

National College of Ireland

Bachelor of Science in Computing

Software Development

Academic Year 2022/2023

Mariusz Kucharski

Student ID: x20113561

x20113561@student.ncirl.ie

“Fits On You”

The e-commerce store

Technical Report

Contents

Document Control.....	5
Executive Summary.....	6
1.0 Introduction	6
1.1. Background	6
1.2. Aims.....	7
1.3. Technology.....	8
1.3.1. ASP.NET and Internet Information Services (IIS) – Main code	8
1.3.2. SQL - Database	9
1.3.3. NodeJS and React.....	10
1.3.4. Python	10
1.3.5. List of all technologies used in the project	11
1.4. Structure	11
2.0 System.....	11
2.1. Requirements.....	11
2.1.1. Functional Requirements.....	12
2.1.1.1. Use Case Diagram (overall view).....	13
2.1.1.2. Use case diagram for a new customer.....	14
2.1.1.3. Use case diagram from the registered customer	16
2.1.1.4. Use case diagram for admin side	20
2.1.2. Data Requirements	24
2.1.3. User Requirements	25
2.1.4. Environmental Requirements	25
2.1.5. Usability Requirements.....	25
2.2. Design & Architecture.....	25
2.2.1. Architectural Diagram.....	25
2.2.2. Database Design.....	27
2.2.3. Class Diagrams	28
2.3. Implementation	29
2.3.1. Main module of the app (ASP.NET)	29
2.3.2. Video-chat module of app (NodeJS, React)	37
2.3.2.1. Backend of video-chat application.....	37
2.3.2.2. Frontend of chat application (client)	42
2.4. Graphical User Interface (GUI).....	55
2.4.1. Client section.....	55
2.4.2. Admin section	61

2.5.	Testing.....	68
2.5.1.	How I performed Unit Tests.....	69
2.5.2.	How I performed Integration Tests.....	77
2.5.3.	How I performed End-to-End Tests.....	95
2.6.	Evaluation	102
3.0	Conclusions	103
4.0	Further Development or Research	104
5.0	References	106
6.0	Appendices.....	108
6.1.	Project Proposal.....	108
6.1.1.	Objectives.....	108
6.1.2.	Background	108
6.1.3.	State of the Art.....	108
6.1.4.	Technical Approach.....	109
6.1.5.	Technical Details	109
6.1.6.	Special Resources Required	110
6.1.7.	Project Plan	111
6.1.8.	Testing.....	113
6.2.	Ethics Approval Application (only if required)	114
6.3.	Reflective Journals	114
6.4.	Invention Disclosure Form (Remove if not completed).....	114
6.5.	Other materials used	114
6.6.	Project Plan	114
6.7.	Backlog.....	115
6.8.	Link to video presentation for Mid-Point of the project.....	116
6.9.	Monthly reports.....	117
6.10.	Link to the FINAL video presentation.....	121

FIGURES

Figure 1.	Use case diagram (overall view)	13
Figure 2.	Use case diagram for unregistered customer.....	14
Figure 3.	Use case diagram for the registered user	16
Figure 4.	Use case diagram for admin side	20
Figure 5.	Architectural diagram of the application (High Level Analysis).....	25
Figure 6.	Database Implementation using Microsoft SQL Server Management Studio.....	27

Figure 7. List of classes used in the project.	28
Figure 8. List of classes used in the project.	29
Figure 9. Class Registration for UI Registration in User Section	30
Figure 10. The part of the code related to the Registration Page	31
Figure 11. Example file structure for my project in ASP.NET with registration highlighting	32
Figure 12. Validation of data entered by the user during registration.....	33
Figure 13. Part of the code in the Category class responsible for connecting the functionality for the product category with the database.....	34
Figure 14. Part of the code responsible for adding a new category to database.....	34
Figure 15. Part of the code responsible for edit and delete a category in database	35
Figure 16. Stored Procedures for Category table	36
Figure 17. The part of script for connecting a database	37
Figure 18. Index.js file from backend of video-chat.....	37
Figure 19. Continue of Index.js file from backend of video-chat.....	38
Figure 20. A fragment of a Context.js file from frontend of video-chat	39
Figure 21. The view of Heroku server on which backend video-chat was deployed.....	39
Figure 22. The view of deploying logs at server Heroku.....	40
Figure 23. Continue of view of deploying logs on the server Heroku.....	41
Figure 24. The running server from a browser view.....	42
Figure 25. App.js file in fronted of video-chat	42
Figure 26. Continue of App.js file in fronted of video-chat	43
Figure 27. Index.js file in fronted of video-chat.....	43
Figure 28. Context.js file in fronted of video-chat	44
Figure 29. Continue of Context.js file in fronted of video-chat	45
Figure 30. Continue of Context.js file in fronted of video-chat	46
Figure 31. A fragment of the code from main application with link to video-chat deployed on Netlify	47
Figure 32. Imports in Context.js file in frontend of video-chat app.....	47
Figure 33. The first component – VideoPlayer.js file	48
Figure 34. The second component – Sidebar.js file	49
Figure 35. Continue of the second component – Sidebar.js file.....	50
Figure 36. The third component – Notifications.js file	51
Figure 37. Development the video-chat by build of the code	52
Figure 38. Netlify – web development platform.....	53

Figure 39. The view of video-chat app already deployed at Netlify	54
Figure 40. Enabling to use camera and microphone by browser	54
Figure 41. Main page of the application	55
Figure 42. Store's offer menu page	56
Figure 43. Login page for registered users.....	57
Figure 44. Registration form for a new customer.....	57
Figure 45. User profile page.....	58
Figure 46. User's purchase history.....	58
Figure 47. Details of the order after click Invoice from purchased history	59
Figure 48. Shopping cart page	59
Figure 49. Order payment – Credit Card option page	60
Figure 50. Order payment – Cash on Delivery option page.....	60
Figure 51. Insight in the invoice after accepting the payment option.....	61
Figure 52. Administration panel – Managing of categories of products page	61
Figure 53. Administration panel – Managing of store's offer and products page.....	62
Figure 54. List of the registered client's page	62
Figure 55. Settings of orders status page	63
Figure 56. The list of information from user's page	63
Figure 57. The printable version of product's list	64
Figure 58. About page.....	64
Figure 59. Contact page with video-chat request option and link to the video-chat app.....	65
Figure 60. View of the Video-chat application.....	66
Figure 61. Display a stock market data	67
Figure 62. Display a stock market information of selected company.....	68
Figure 63. Workflow file (config.yml).....	99
Figure 64. Example of CI/CD simple Pipelines workflow	100
Figure 65. Built and testing details for pipeline (1/2)	101
Figure 66. Built and testing details for pipeline (2/2)	102
Figure 67. Planned tasks to be carried out as part of the project development.....	114
Figure 68. Tasks and milestones visualized on a timeline.	115
Figure 69. Backlog representing parts of tasks to do.....	115

TABLES

Table 1. User story tasks in terms of use case diagram for unregistered customer14

Table 2. User story tasks in terms of use case diagram for the registered user17

Table 3. User story tasks in terms of use case diagram for the administrator.....22

Document Control

Distribution List

Name	Title
Mariusz Kucharski	Student ID: x20113561 Specialization: Software Development
Frances Sheridan	Lecturer and Supervisor
William Clifford	Project supervisor

Important links

Purpose	Links
GitHub Project code repositorium	https://github.com/bregor77/FitsOnYou
Link to final video presentation	https://youtu.be/kk5kHgoXeC8
Link to LinkedIn	https://www.linkedin.com/in/mariuszkucharski/

Executive Summary

The "Fits on You" project's implementation will be examined in this report. From the fundamental features that let users create accounts and log in to them to browsing, searching, and purchasing products available in the shop. The system's extra features, which include features of the application like video chat with a store professional and stock market information display, will also be highlighted. The desired outcomes of the report should include a thorough breakdown of the system's correct functionality as well as potential future areas for improvement to make the system easier to use. The suggested project structure entails decomposing each functionality into the essential elements that it will offer to the user and outlining the underlying technology that will be used to support that functionality. The expected report outcomes will not be the actual report findings, but the main objective is to include as many of the actual project results as feasible in the finished product. Changes from new ideas and upgrades to existing ones are anticipated as the project life cycle progresses.

1.0 Introduction

1.1. Background

One of the intentions of the idea for my project was to make the application, seemingly one of many, significantly stand out from the others. In the field of e-commerce applications, it is important to interest and attract the attention of both the customer of such a store and the interest of potential e-commerce store owners in the application. In this matter, not only the price of the product is important, but also an innovative approach to customer needs. Here, the most important elements that distinguish my application from the others are the possibility of video consultation with a store expert and displaying stock market information.

Based on recent years, it can be seen that people's preferences are changing. Once the average customer preferred to go to the store himself, but now he prefers to do many things online. Thanks to this form of shopping, people feel safer through the comfort of shopping while staying at home. This also saves time that people spend shopping in a traditional store, where the selection of available goods is also smaller. [1] However, this type of shopping makes it more difficult to decide which product to choose, as a product from the customer's perspective is just an image on the monitor screen and it is sometimes difficult to assess whether it is suitable for the customer. [2] As my application will concern a store related to the sale of clothes, thanks to the use of video consultations, the customer will be able to contact the store's department, where a stylist or other specialist will be able to advise on the selection of appropriate clothes through a videoconference built into my application. The video conference included in the sales application is not only a convenient addition for the customer, but also increases the prestige of the store, which in such a situation is perceived as more professional.

The user who browses the website is not always motivated to look for additional information in social media, which is why video chat meets the customer's expectations here. The built-in video-chat function gives the client active support tailored to the client's needs. At this point, it could be said that live-chat-bots perform such tasks equally well. Well, not

entirely, because despite the obvious advantages of chat-bots, it is hard to imagine full customer service without the human factor. In many cases, only the function of the chat-bot built into the application and the inability to solve all problems can frustrate the customer who needs help even more, thus reducing the general interest in shopping in such a store and weakening sales statistics.

Based on socio-demographic research, customers nowadays are more impatient and do not want to describe the problem by e-mail and wait for a response. It is similar when it comes to contact forms, which do not fulfil their role, because you wait for hours for a response if there is one at all. Video-chat, through direct interaction with a store representative, offers immediate help in solving these types of problems. If the customer has any questions or dilemmas, they will be able to address them directly, which in turn may contribute to choosing our online store over the competition.

Thanks to the stock market data integrated with the e-commerce application, the store using my application will be able to provide its customers with up-to-date information about shares that can help them make informed investment decisions. Integrating unique features such as real-time stock market data and video conferencing with store experts can differentiate the e-commerce application from competitors and provide added value to customers.

Overall, creating a well-designed, user-friendly, and secure e-commerce application can be a significant investment, but it has the potential to open up new revenue streams and expand the customer base for businesses of all sizes.

1.2. Aims

My project aims to create an application that will allow you to make purchases in an easy and pleasant way, which will make it a good idea to attract more customers, by what make the application a sought-after product for people and companies who want to open new businesses or increase the sale of your products on the Internet. This is to attract the most demanding and undecided customers to the store and encourage them to buy.

The store meets customer expectations. Due to the fact that video-chat is a service not based on currently known and common applications such as Zoom, Skype, Microsoft Teams, etc., there are no threats of service shutdown, interruption, time and financial restrictions, additional costs, or increased costs. Video-chat is an integral part of the application, and this service is managed directly by the store owner.

Another important assumption and aspect affecting competitiveness in comparison with other known applications offering video-chat services is the simplicity of operation and the fact that the customer is not obliged to provide its data, log in or configure his device in advance to use the video chat application. This is another advantage affecting the overall attractiveness of the company on the market and encouraging the customer to use the services. Thanks to this way of communication, we have the possibility of an interactive and personalized conversation that takes place in real time.

This type of simple messenger allows the customer to ask questions that bother him while browsing the site.

- help in choosing a product
- assistance in order fulfilment
- assistance with complaints or returns

1.3. Technology

1.3.1. ASP.NET and Internet Information Services (IIS) – Main code

Choosing the right technology is the key to the success of the entire project. My choice of technology is dictated by the client's needs and the specificity of the business, which in my case is online sales. I create my project based on ASP.NET, which is a set of technologies based on a framework designed by Microsoft. [3] For this purpose, I use the free development environment Visual Studio Community Edition 2022. [4]

ASP.NET (Active Server Pages.NET) is an environment that allows you to easily create dynamic Internet applications, home pages or online stores. It connects various kinds of elements of the WWW. The technology was developed by Microsoft. [5]

ASP.NET is the latest version of the well-known ASP technology. Its new version is very different from the classic one. The ASP.NET programming model is transformed and improved and includes many new tools that make it much easier to design and create dynamic web pages. One of the benefits of this technology is that it provides powerful tag controls. They are very similar to HTML tags. ASP.NET provides many new possibilities and extensions, it allows you to display randomly selected ads or a calendar. Thanks to the control elements, the programmer can generate complex HTML code corresponding to the applicable standards. The work required for this activity is minimal. The ASP.NET technology allows you to use the information contained in databases and allows you to adjust the website depending on the preferences of the users who visit it. [6]

Currently, ASP.NET offers three frameworks for building web applications: Web Forms, ASP.NET MVC (Model-View-Controller) and ASP.NET Web Pages. All three platforms are stable and perfect for web application development. For my purposes, I believe that the Web Forms version will be optimal. With ASP.NET Web Forms, you can create dynamic websites using the familiar drag-and-drop event model. [7]

ASP.NET and Internet Information Services (IIS) are two powerful technologies that offer several advantages for web development and hosting. [8] [9] One significant advantage of using ASP.NET and IIS Express is scalability. These technologies can handle high volumes of traffic and are capable of handling thousands of requests per second. This makes them ideal for large web applications that require scalability. Another advantage of ASP.NET and IIS is security. They have built-in security features that help protect against common web vulnerabilities like cross-site scripting (XSS), SQL injection, and cross-site request forgery (CSRF). Additionally, they support SSL encryption, which provides secure communication between the server and client. Developers can also benefit from using ASP.NET and IIS because of the easy development process. ASP.NET provides a wide range of tools and libraries for building web applications, making it easier for developers to create dynamic,

responsive, and user-friendly web applications. IIS provides an easy-to-use interface for managing websites, application pools, and other server configurations. Another advantage of using ASP.NET and IIS is compatibility. ASP.NET is compatible with different programming languages, including C#, VB.NET, and F#, making it easier for developers to work with existing codebases and integrate with other platforms and technologies. Performance is another area where ASP.NET and IIS excel. They are optimized for high performance and fast response times. Features such as caching, precompilation, and just-in-time (JIT) compilation help speed up the execution of web applications. [10] [11] [12]

Finally, ASP.NET and IIS are stable and reliable technologies that have been extensively tested and are widely used in production environments. Overall, the combination of ASP.NET and IIS provides a powerful platform for web development and hosting, with several advantages in scalability, security, easy development, compatibility, performance, and reliability. [13]

1.3.2. SQL - Database

The database is created using Microsoft SQL Server Management Studio and get stored on Microsoft SQL Server. Using a SQL-based database created in Microsoft SQL Server with ASP.NET technologies offers several benefits. SQL Server is a scalable database system that can handle large volumes of data and multiple concurrent users, making it ideal for web applications. It is designed for high performance and fast query response times, using advanced indexing and caching techniques to optimize data retrieval and processing.

SQL Server provides robust security features to protect sensitive data, including encryption, role-based access control, and auditing. It integrates seamlessly with ASP.NET technologies, including the Entity Framework, which allows developers to work with databases using object-oriented programming techniques. [14]

Microsoft SQL Server Studio provides an easy-to-use interface for managing databases, tables, and queries, making it easy for developers to create and manage databases. SQL Server is compatible with a wide range of programming languages and platforms, including ASP.NET, making it easy to integrate with other technologies. [15]

One of the great advantages of using Microsoft SQL database is the use of so-called System Stored Procedures, created and edited using SQL Server Management Studio. System stored procedures in Microsoft SQL Server are pre-defined, precompiled routines that are an integral part of the SQL Server system. [16] For my project, System Stored Procedures are predefined, precompiled routines that provide a convenient and efficient way to perform CRUD (Create, Read, Update, Delete) operations on the database. These procedures are designed to handle common data manipulation tasks and provide a standardized approach to performing CRUD operations. They encapsulate the logic required for these operations, ensuring consistency, and reducing the likelihood of errors in the code. System Stored Procedures are precompiled and optimized by the SQL Server engine. As a result, they can offer better performance compared to dynamically constructed SQL queries, especially for complex CRUD operations. When using system stored procedures, only the procedure call and parameters need to be sent over the network, rather than entire SQL statements. This reduces network traffic and improves application responsiveness, particularly in distributed systems. Since the logic for CRUD operations resides in stored procedures, any modifications

or updates can be made centrally within the database. This simplifies maintenance and version control, as changes do not require recompiling application code. By keeping CRUD operations in Stored Procedures, I could focus on implementing application-specific business logic in the application code. This separation enhances code readability and maintainability. [17] [18]

1.3.3. NodeJS and React

Using Node.js and React for video chat offers several advantages that make them a great choice for building such applications.

Node.js is known for its event-driven, non-blocking I/O model, making it well-suited for handling real-time communication. Video chat requires low-latency communication, and Node.js excels in managing multiple concurrent connections efficiently. It has robust support for WebSockets, a crucial technology for enabling bi-directional communication between the server and clients. This is fundamental for establishing and maintaining real-time video streams in a video chat application. Node.js's single-threaded event loop, coupled with its ability to handle asynchronous requests, makes it highly scalable. In a video chat application where numerous users may be connected simultaneously, scalability is essential to ensure a smooth user experience.

On the client-side, React, a popular JavaScript library, offers a component-based architecture. This allows developers to build UI components independently and compose them to create complex user interfaces. This modularity simplifies the development process and helps maintain a clean codebase. React's Virtual DOM efficiently updates and renders only the necessary components, reducing the computational overhead and improving performance. This is crucial for real-time video chat, where constant updates are needed to maintain a smooth video stream. React can easily integrate with WebRTC (Web Real-Time Communication), a collection of APIs that enable peer-to-peer communication between browsers. WebRTC is essential for establishing direct video streams between users in a video chat application. [19] [20]

Combining Node.js and React provides a powerful and efficient stack for building real-time video chat applications that deliver a smooth and responsive user experience.

1.3.4. Python

Python is considered one of the best choices for performing stock market analysis. Python boasts a plethora of powerful libraries such as Pandas, NumPy, and Matplotlib, which are widely used for data manipulation, numerical computing, and data visualization. These libraries simplify the handling of financial data and enable analysts to perform complex calculations efficiently. Python's Pandas library provides data structures like DataFrame, which is ideal for organizing and analysing structured financial data. Analysts can easily clean, filter, and transform data, allowing them to gain valuable insights from historical market data.

Integration and Flexibility: Python's versatility and compatibility with other languages make it easy to integrate with various financial platforms and tools. It can seamlessly connect with brokerage APIs, financial databases, and third-party services, providing analysts with diverse data sources.

Visualization Capabilities: Python's Matplotlib, along with libraries like Seaborn and Plotly, facilitates the creation of insightful visualizations. Effective data visualization helps analysts understand patterns, trends, and anomalies in the market data. [21] [22]

1.3.5. List of all technologies used in the project

- **ASP.NET** – Main code of the app
- **SQL** - Database
- **NodeJS** – Video Chat
- **REACT** – Video Chat
- **Python** – Stock Market Data retrieve and analysis

1.4. Structure

The structure of the technical report regarding the application is divided into three main categories, including:

- 1) **Requirements:** This section provides a detailed breakdown of the systems' functional and non-functional needs, as well as a use case diagram.
- 2) **Design and Architecture:** This part includes several diagrams of the system architecture as well as details regarding the system flow.
- 3) **GUI:** Displays screenshots of the application's development process.

2.0 System

2.1. Requirements

To create a successful e-commerce application, there are several requirements that should be considered, including:

- ✓ **User-friendly design:** The e-commerce application should have an intuitive and easy-to-use design that allows customers to browse and purchase products quickly and easily.
- ✓ **Secure payment processing:** Payment processing must be secure to protect customer data and prevent fraud. Implementing trusted payment gateways and SSL encryption can help to ensure security.
- ✓ **Mobile optimization:** The e-commerce application should be optimized for mobile devices, including smartphones and tablets, to provide a seamless shopping experience on-the-go.
- ✓ **Integration with inventory management system:** The e-commerce application should be integrated with the business's inventory management system to ensure accurate stock levels and prevent overselling.
- ✓ **Integration with shipping providers:** The e-commerce application should integrate with shipping providers to provide real-time shipping rates, tracking information, and delivery options.
- ✓ **Customer support features:** The e-commerce application should have customer support features such as live chat, email support, and phone support to help customers with questions and issues.

- ✓ **Analytics and reporting:** Data analytics should be incorporated into the e-commerce application to track user behaviour, measure performance, and identify areas for improvement.
- ✓ **SEO optimization:** The e-commerce application should be optimized for search engines to increase visibility and attract more traffic.
- ✓ **Integration with social media:** The e-commerce application should be integrated with social media platforms to provide a seamless shopping experience and enable customers to share their purchases with their network.
- ✓ **Application features:** The e-commerce application should be designed to provide a seamless, secure, and personalized shopping experience for customers while maximizing revenue opportunities and minimizing operational costs for the business.
- ✓ **Unique features:** Offering unique features such as real-time stock market data or video conferencing with store experts can differentiate the e-commerce application from competitors and provide added value to customers.

2.1.1. Functional Requirements

In this section, instead of a traditional use case description, I used the form of a user story known from agile software development. [23] [24] There are certain similarities between user stories and use cases, such as the action being taken by the object or user, the events that should follow, and outcome of the action. A user story gives a succinct description of one or more software needs, including their who, what, and why. Developers can concentrate on perspectives, features, functionality, and results by using user stories to provide context for interactions. [25] [26] [27]

2.1.1.1. Use Case Diagram (overall view)

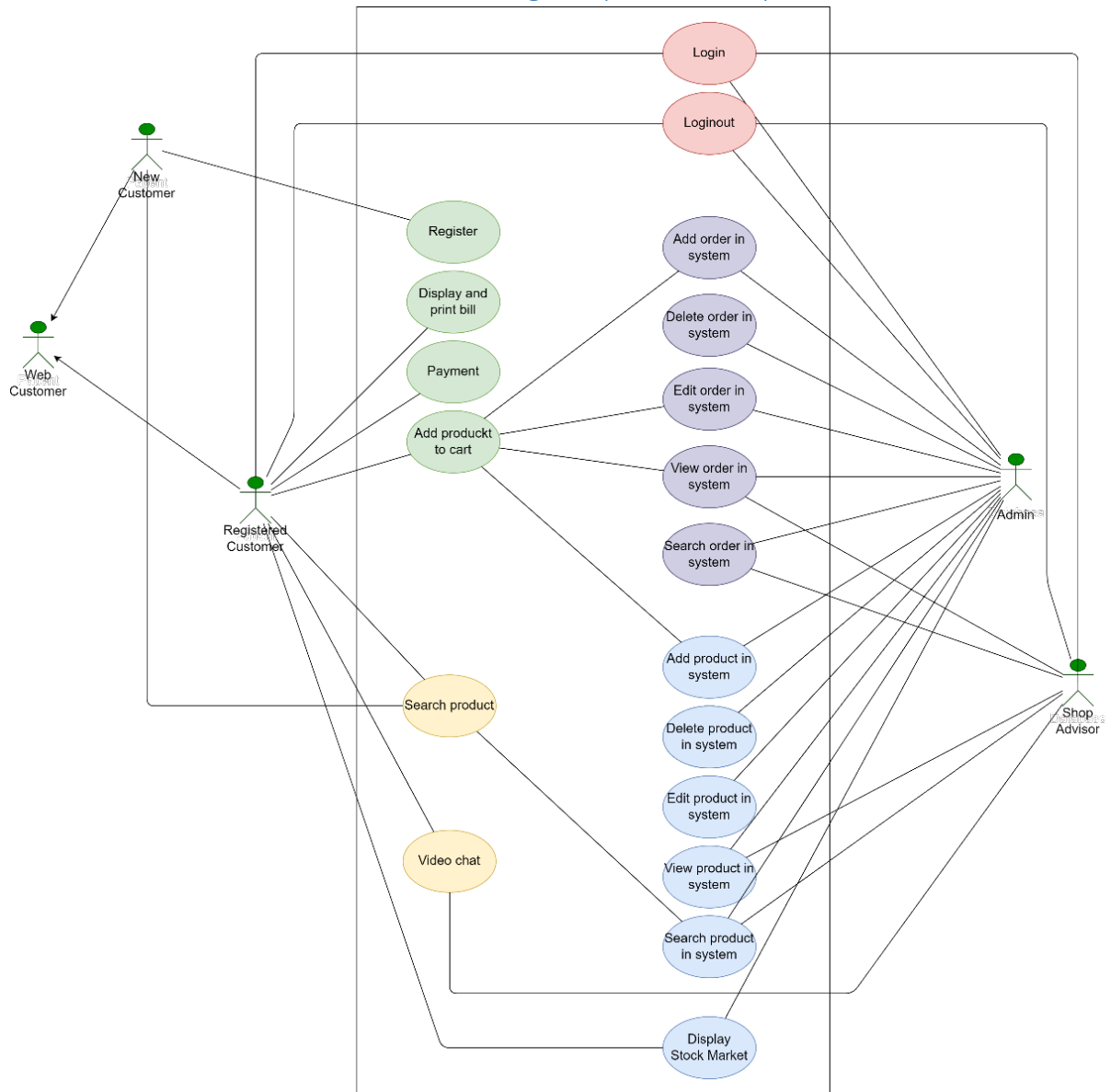


Figure 1. Use case diagram (overall view)

2.1.1.2. Use case diagram for a new customer

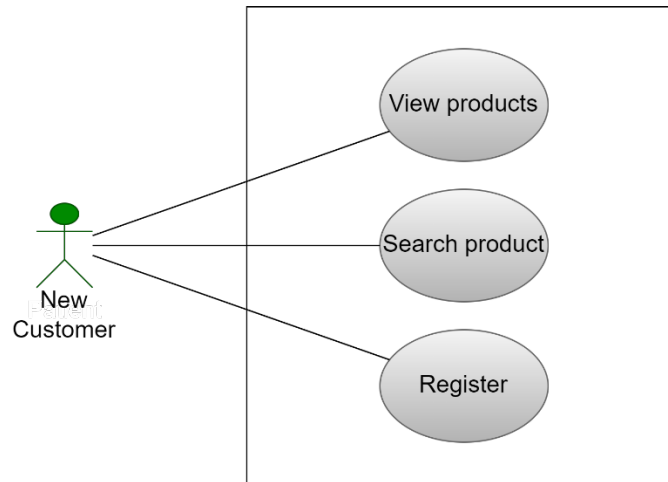


Figure 2. Use case diagram for unregistered customer

User story ID	User story	Acceptance criteria	Story point
US1	As an unregistered customer, I want to view the store's offer.	When an unregistered user opens the app, the product categories page will appear. The new user will be able to choose the product category that interests him. The new user will be able to browse products within the previously selected category.	2
US2	As an unregistered customer, I want to search the store's offer.	The new user will be able to search for the product he is interested in.	2
US3	As an unregistered customer, I want to register.	After clicking on the appropriate registration button, the new user will be able to register on a separate page for registering new users.	2

Table 1. User story tasks in terms of use case diagram for unregistered customer

Tasks

US1:

1. Create a UI for the main page of the store that included category of the products.
2. An unregistered user is able to view a store's category of product offer.
3. Create a UI for the products page that contains a store's offer of products.
4. An unregistered is able to view products offer in the in the products tab after selecting category of products.
5. Create a test case to satisfy the acceptance criteria. Products category, products as well as products details should be displayed in the appropriate tabs.

US2:

1. Create a UI for the page included search field.
2. Search the database in the products table.
3. Display on the products page the product list of items matching the query and phrase entered by the user.
4. Display a detail of selected product (including product description, product image and price).
5. Create a test case to satisfy the acceptance criteria. This included attempt to search product that exists as well as products that do not exist in the database.

US3:

1. Create a UI for the registration page form
2. Store user data in a database (including login information)
3. Create an authentication mechanism
4. Verify user data already exists during registration and if attempting to login
5. Verify email/password satisfies minimum criteria (capitals, digits, special character, number of letters)
6. Create a test case to satisfy the acceptance criteria. This includes creating a user with valid entry types in all fields of the registration form, as well as attempting to create a new user with illegal entry types in the form

2.1.1.3. Use case diagram from the registered customer

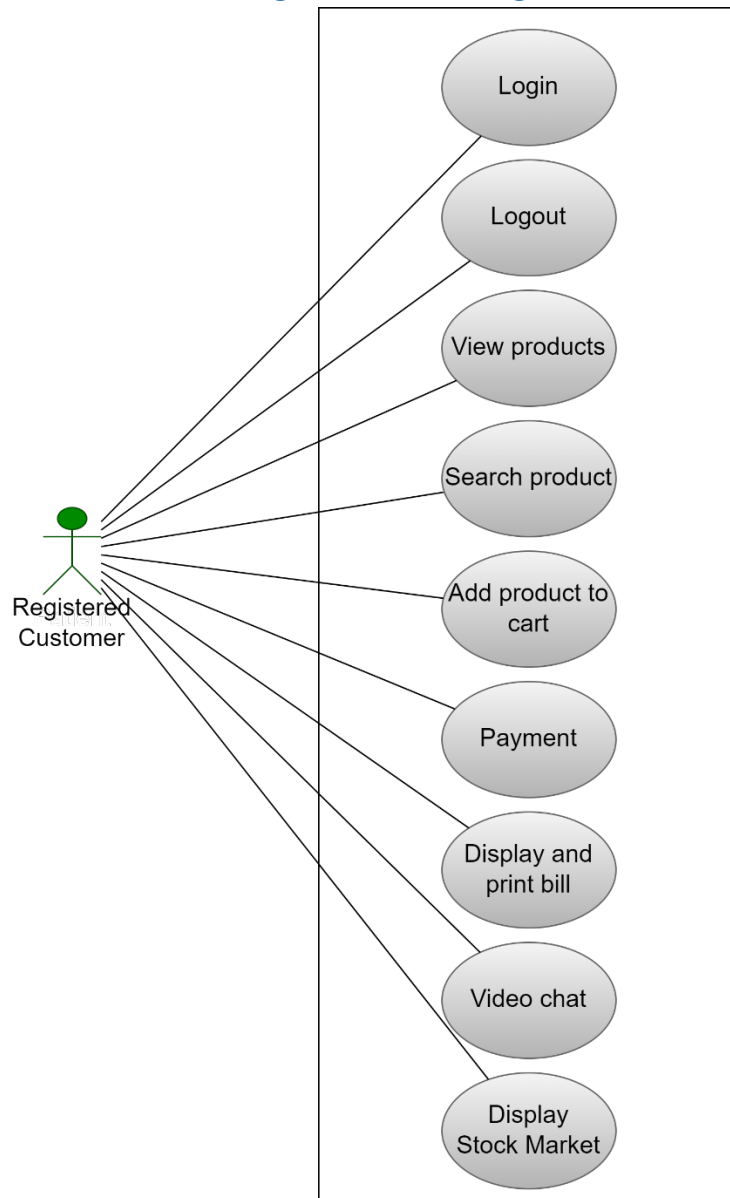


Figure 3. Use case diagram for the registered user

User story ID	User story	Acceptance criteria	Story point
US1	As a registered user, I want to login so that I have a personalized experience on the application.	When the user enters their email and password and is greeted by their personalized page then this is satisfied.	1
US2	As a registered user, I want to be able to log out so that no one can use my login profile when I finish working with the application.	When the user enters their email and password and is greeted by their personalized page then this is satisfied.	1

US3	As a registered user, I want to view the store's offer.	When an unregistered user opens the app, the product categories page will appear. The new user will be able to choose the product category that interests him. The new user will be able to browse products within the previously selected category.	2
US4	As a registered user, I want to search products.	After the user enters the appropriate product name in the search field, the page displays the list of products that match the search phrase. After clicking on the indicated product, they are display details about that product if the product is in the database.	2
US5	As a registered user, I want to add a product to the cart.	The registered user is able to add selected product to the shopping cart. The list of products can be edited before purchase or deleted which means that the shopping process will be cancelled. To continue shopping, user can add the new products to the shopping cart.	3
US6	As a registered user, I want to make a payment.	The payment is made by implementing the API and pop-up windows with external services such as PayPal, Revolut, bank services for card payments.	3
US7	As a registered user, I want to display and print a bill.	By pressing the Print button, the purchase invoice is printed.	3
US8	As a registered user, I want to establish video chat with store expert.	By pressing the Video Chat button, the registered user is transferred to a new window where you can schedule a meeting or start a video chat with a store specialist.	3
US9	As a registered user, I want to display a stock market.	By pressing the Stock Market button, a new subpage with stock market information opens.	3

Table 2. User story tasks in terms of use case diagram for the registered user

Tasks

US1:

1. Create a UI for the login page that included a form for user logging in by entering e-mail address and password.
2. Only registered user is able to login to the system.
3. The information entered by the user is validated at login to ensure that the email address and password are in the correct format.
4. If the user is already registered and of login data entered by the user will coincide with the credentials in store's customer database, the user will be logged in.

US2:

1. Create a UI for the logout page that included a button used to log the user out of the system.
2. Only the currently logged in user can log out.
3. A logged-out user still has access to browsing the store's offer.
4. A logged-out user cannot make a purchase (there is no option to add to cart and pay).
5. A logged-out user can log back in.

US3:

1. Create a UI for the main page of the store that included category of the products.
2. A registered user is able to view a store's category of product offer.
3. Create a UI for the products page that contains a store's offer of products.
4. A registered user is able to view products offer in the in the products tab after selecting category of products.
5. Create a test case to satisfy the acceptance criteria. Products category, products as well as products details should be displayed in the appropriate tabs.

US4:

1. Create a UI for the page included search field.
2. Search the database in the products table.
3. Display on the products page the product list of items matching the query and phrase entered by the user.
4. Display a detail of selected product (including product description, product image and price).
5. Create a test case to satisfy the acceptance criteria. This included attempt to search product that exists as well as products that do not exist in the database.

US5:

1. Create a UI for the page included a view of shopping cart and payment option.
2. The list of selected items is displayed in the "Selected items for purchase" section.
3. The user has the option of verifying his profile and shipping address before making a payment selection.
4. The list of selected items and displayed in the shopping cart can be modified before choosing the type of payment.

US6:

1. By pressing the "Continue shopping" button on the UI for page included shopping cart and list of selected products, opens UI that included a window with shipping options.
2. By pressing the "Make a payment" button, opens a window with a choice of payment options. The payment is made by implementing the API and pop-up windows with external services such as PayPal, Revolut, bank services for card payments.
3. The customer has the option of resigning from purchases at any time before making the payment by pressing the "Cancel order" button on the UI that included shopping cart.

US7:

1. Create a UI for the page containing the purchase confirmation and buttons for printing the invoice.
2. An invoice can be printed in the form of a printout using a hardware printer available in the user's system or a printout in the form of a PDF or CSV file.

US8:

1. Create a UI for the page included a page for establishing videoconference calls with store specialist and advisor.
2. The videoconferencing connection can be scheduled, or the customer can wait in line for the connection to be made during certain working hours of the store advisor.

US9:

1. Create a UI for the page included a page to display various types of stock market information depending on the user's request.
2. The stock market information screen is displayed on the page in the form of pop-up windows.
3. The stock market information analyse can be printed in the form of a printout using a hardware printer available in the user's system or a printout in the form of a PDF or CSV file.

2.1.1.4. Use case diagram for admin side

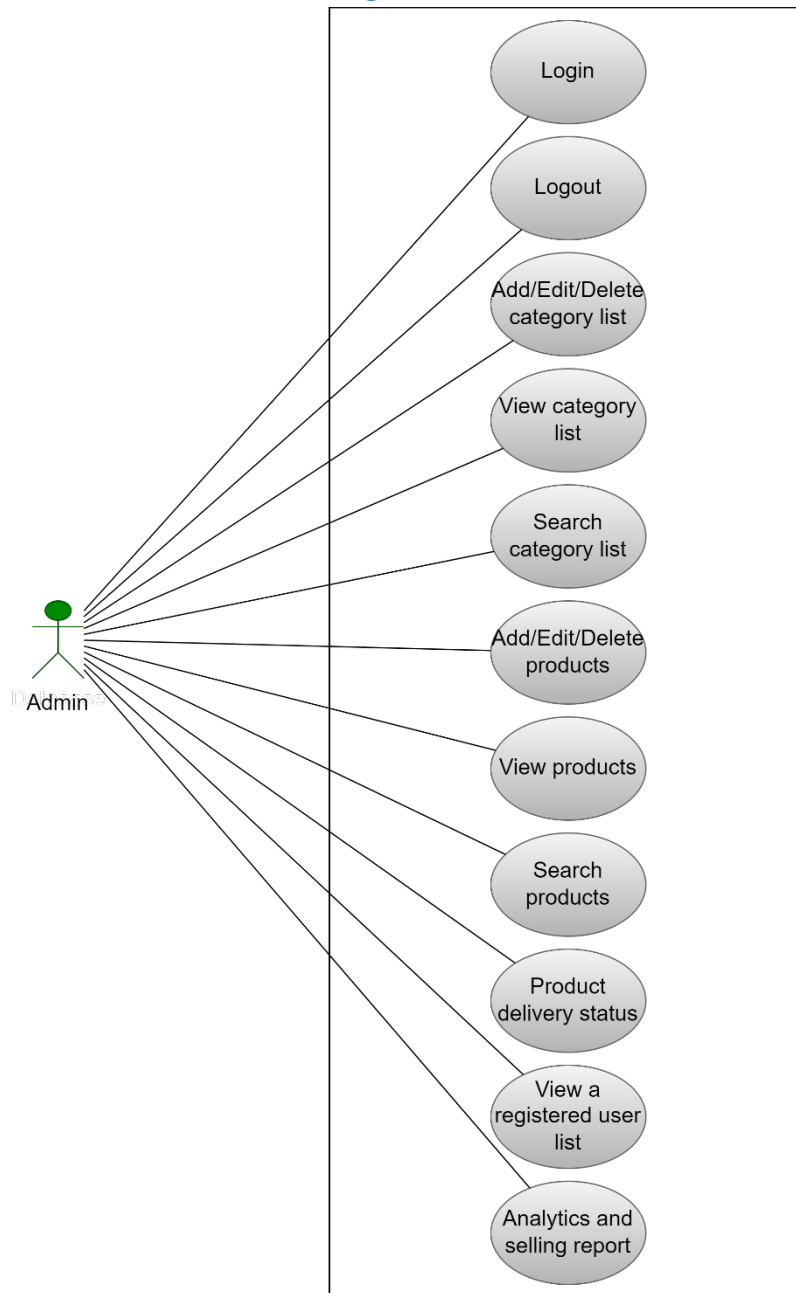


Figure 4. Use case diagram for admin side

User story ID	User story	Acceptance criteria	Story point
US1	As an admin, I want login to administration panel so that I can have special privileges to interact with the system, that other users wouldn't otherwise have.	When the administrator can login through our login page and see specialized buttons then we know the story has been satisfied.	1

US2	As an admin, I want to be able to log out so that no one can use my login profile when I finish working with the application in administration panel.	When the administrator enters their email and password and is greeted by their personalized page then this is satisfied.	1
US3	As an admin, I want to manage the products categories in administration panel. This includes creating, editing, and deleting of product categories, includes managing category, descriptions, images, and prices.	When the administrator can manage (add/edit/delete) product categories through a form specially created for this purpose, we know that the story has been satisfied.	2
US4	As an admin, I want to view in administration panel the category list of products.	We know the story has been satisfied when the admin can view the list of product categories.	2
US5	As an admin, I want to search product categories in the administration panel.	We know that the story has been satisfied when the admin can see the results of the searched product categories that match the search phrase entered in the corresponding search box.	2
US6	As an admin, I want to manage the products that are available for sale on the website. This includes creating, editing, and deleting products, as well as managing product descriptions, images, and prices.	When the administrator can manage (add/edit/delete) products through a form specially created for this purpose, we know that the story has been satisfied.	2
US7	As an admin, I want to view products in the administration panel.	We know the story has been satisfied when the administrator can view the list of products.	2
US8	As an admin, I want to search products in admin panel.	We know that the story has been satisfied when the administrator can see the results of the searched products that match the search phrase entered in the corresponding search box.	2
US9	As an admin, I want to be able to manage the shipping options that are available on the website. This includes delivery	When the admin will be able to modify the shipping options and see the delivery status by displaying this	2

	status, setting up shipping carriers, managing shipping rates, and resolving any shipping-related issues.	information in a specially created tab for settings, shipping options preview and orders, we know that the story has been satisfied.	
US10	As an admin, I want to be able to manage customer accounts and profiles. This includes viewing customer details, managing their orders, and resolving any customer issues.	When the administrator can view and options related to customer accounts profiles, we know that the story has been satisfied.	3
US11	As an admin, I want to be able to view detailed reports and analytics that provide insights into the store's performance. This includes tracking sales, revenue, and customer behaviour, and analysing trends and patterns.	When the administrator will be able to view detailed sales reports and analyses in the Selling Reports tab created especially for this purpose, we know that the story has been satisfied.	3

Table 3. User story tasks in terms of use case diagram for the administrator

Tasks

US1:

1. Create a UI for the login page in admin panel section of application that included a form for admin logging in by entering e-mail address and password.
2. Only a user with administrator privileges currently existing in the database can log in to the administration panel. The first administrator is created by the application developer. An administrator login details are included in the contract and safely delivered by the developer to the downstream store owner.
3. The information entered by the user is validated at login to ensure that the email address and password are in the correct format.
4. Create a test case to satisfy the acceptance criteria. Only logged in user with admin credentials has access to the administration panel.

US2:

1. Create a UI for the logout page that included a button used to log the user out of the system.
2. Only the currently logged in admin can log out.
3. A logged-out administrator can log back in.
4. Create a test case to satisfy the acceptance criteria. Admin is able to logout from the administrator panel to prevent against unwanted access by third parties and keep a safe access to the system after finishing work.

US3:

1. Create a UI for the page of the administrator panel that includes product categories management.
2. Only a user registered as an administrator and logged in to the administration panel has the ability to manage the store's product categories.
3. All actions entered by the administrator in the administration panel regarding product categories are reflected in the database.
4. Create a test case to satisfy the acceptance criteria. Product categories list can be added, modified, and deleted by administrator.

US4:

1. Create a UI for the page of the administrator panel that includes ability to view product categories list.
2. Only a user registered as an administrator and logged in to the administration panel has the ability to view the store's product category.
3. Create a test case to satisfy the acceptance criteria. Products category list is displayed in the appropriate category list section.

US5:

1. Create a UI for the page of the administrator panel that includes the ability to search for product categories from the list of categories.
2. Only a user registered as an administrator and logged in to the administration panel has the ability to view the store's product category.
3. Create a test case to satisfy the acceptance criteria. Products category list requested in the search field and found in the database is displayed in the appropriate category list section.

US6:

1. Create a UI for the page of the administrator panel that includes product offer management.
2. Only a user registered as an administrator and logged in to the administration panel has the ability to manage the products offered in the store.
3. All actions entered by the administrator in the administration panel regarding products are reflected in the database.
4. Create a test case to satisfy the acceptance criteria. The products can be added, modified, and deleted by administrator.

US7:

1. Create a UI for the page of the administrator panel that includes ability to view products list.

2. Only a user registered as an administrator and logged in to the administration panel has the ability to view products offered in the store.
3. Create a test case to satisfy the acceptance criteria. A product list is displayed in the appropriate product list section.

US8:

1. Create a UI for the page of the administrator panel that includes the ability to search for products offered in the store.
2. Only a user registered as an administrator and logged in to the administration panel has the ability to view the store's products.
3. Create a test case to satisfy the acceptance criteria. Products list requested in the search field and found in the database is displayed in the appropriate product list section.

US9:

1. Create a UI for the page of the administrator panel that includes the ability to manage the shipping options, delivery status, setting up shipping carriers, managing shipping rates.
2. Management of all shipping options and delivery status preview is carried out by using the appropriate buttons in the administration panel in the Delivery status section.
3. Create a test case to satisfy the acceptance criteria includes ability to successfully manage all actions listed in the US9 point 1.

US10:

1. Create a UI for the page of the administrator panel that includes the ability to manage customer accounts and profiles.
2. Create a test case to satisfy the acceptance criteria includes ability to successfully manage all actions listed in the US10 point 1.

US11:

1. Create a UI for the page of the administrator panel that includes the ability to able to display detailed reports and analytics.
2. Create a test case to satisfy the acceptance criteria includes ability to successfully manage all actions listed in the US11 point 1.

2.1.2. Data Requirements

To view detailed reports and analytics of various information of stock market I used external sources of stock market information and attached them as an API in the application code for the section related to the analysis of stock market information.

2.1.3. User Requirements

From the user's standpoint, the web application must display store offer and allow quick, simple ordering products. All visual representations must be easily accessible, and several types of information illustrating potential website activities must be included in the final product. Before placing an order, the user must first register with the app.

2.1.4. Environmental Requirements

Any web browser and any internet-connected device can access and view the web application.

2.1.5. Usability Requirements

The "Fits on you" web application includes a simple user interface that makes it simple for customers to get to the ordering page from any page. It offers information on the admin side features and makes it simple to edit everything without a background in coding.

2.2. Design & Architecture

2.2.1. Architectural Diagram

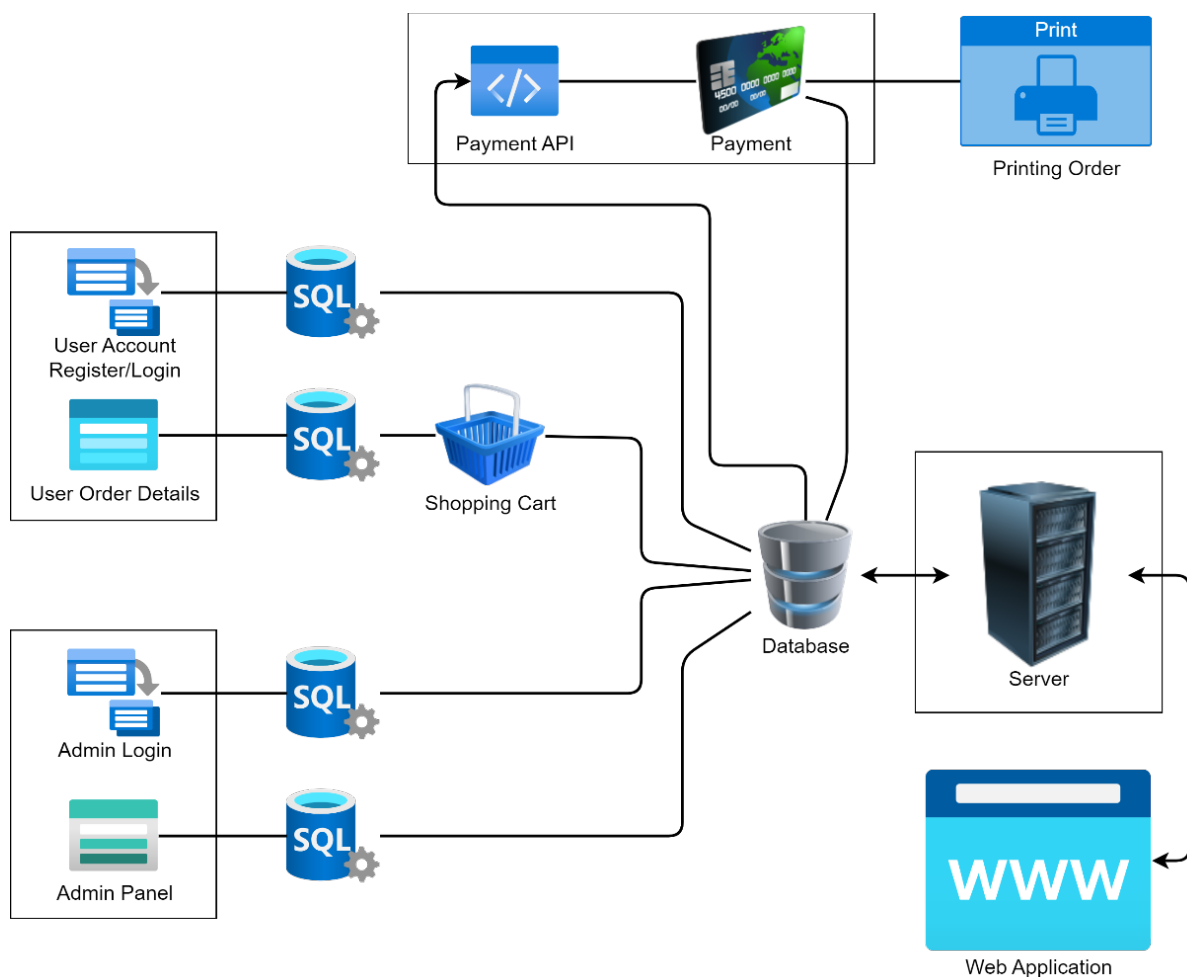


Figure 5. Architectural diagram of the application (High Level Analysis)

In the diagram, we can distinguish two sections for the customer and for the administrator.

The potential customer has access to the section containing the shop offer. As an unregistered customer, has the option of registering, and then, as a registered customer, he has the option of making purchases by adding products to the shopping cart, making payment, and printing an invoice. A user can add a product to the cart by clicking a button. The user is to be able to see the shopping cart summary. All the products that have been added to the shopping cart by the user are listed along with their price and the quantity. The total price of all the products added to cart is displayed. A user can edit the quantity of each product or remove the product from the shopping cart. A user can remove the product from the cart by clicking a button or by dragging the product and dropping it outside the cart. The total price changes accordingly when a user edits the quantity of a product or when a product is removed from the cart.

The second section of my app is the administration panel and is designed to manage all elements of the virtual store. The admin panel can be accessed by the store owner, or a person authorized to manage all elements of the virtual store through the login process. The admin has tools to manage the store offer and product categories. This includes creating, editing, and deleting categories and products, as well as managing product descriptions, images, and prices. The admin also has tools to manage shipping options, set up carriers, manage shipping costs, and troubleshoot any shipping issues. The admin also has access to various types of detailed reports and analytics that give insight into the performance of the store. All this information is reflected in the database and from this place you can make all kinds of changes regarding the management of the online store.

All data regarding accounts, both customers and administrators, products, lists of product categories are stored on Microsoft SQL Server. On the server, the database also stores information related to the history of purchases and a complete list of all orders. This is needed to create reports for the store owner's sales analysis.

2.2.2. Database Design

The design of the database was similar to the analysis phase. The database has been developed using Microsoft SQL Server Management Studio 19.0.1.

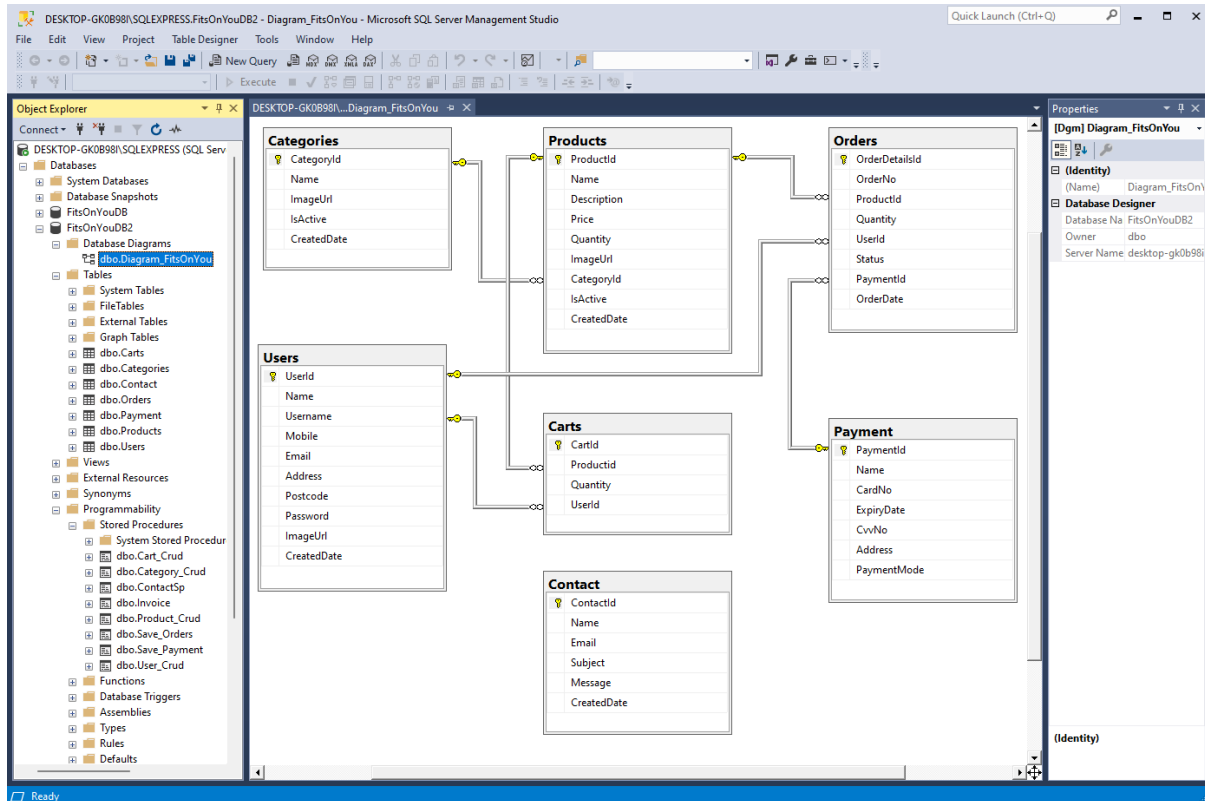


Figure 6. Database Implementation using Microsoft SQL Server Management Studio

These are the main tables in the application and others are lookup and query tables. The tables were derived from the ER-Diagram. The diagram provides a visual representation of the entities in the database and their relationships.

2.2.3. Class Diagrams

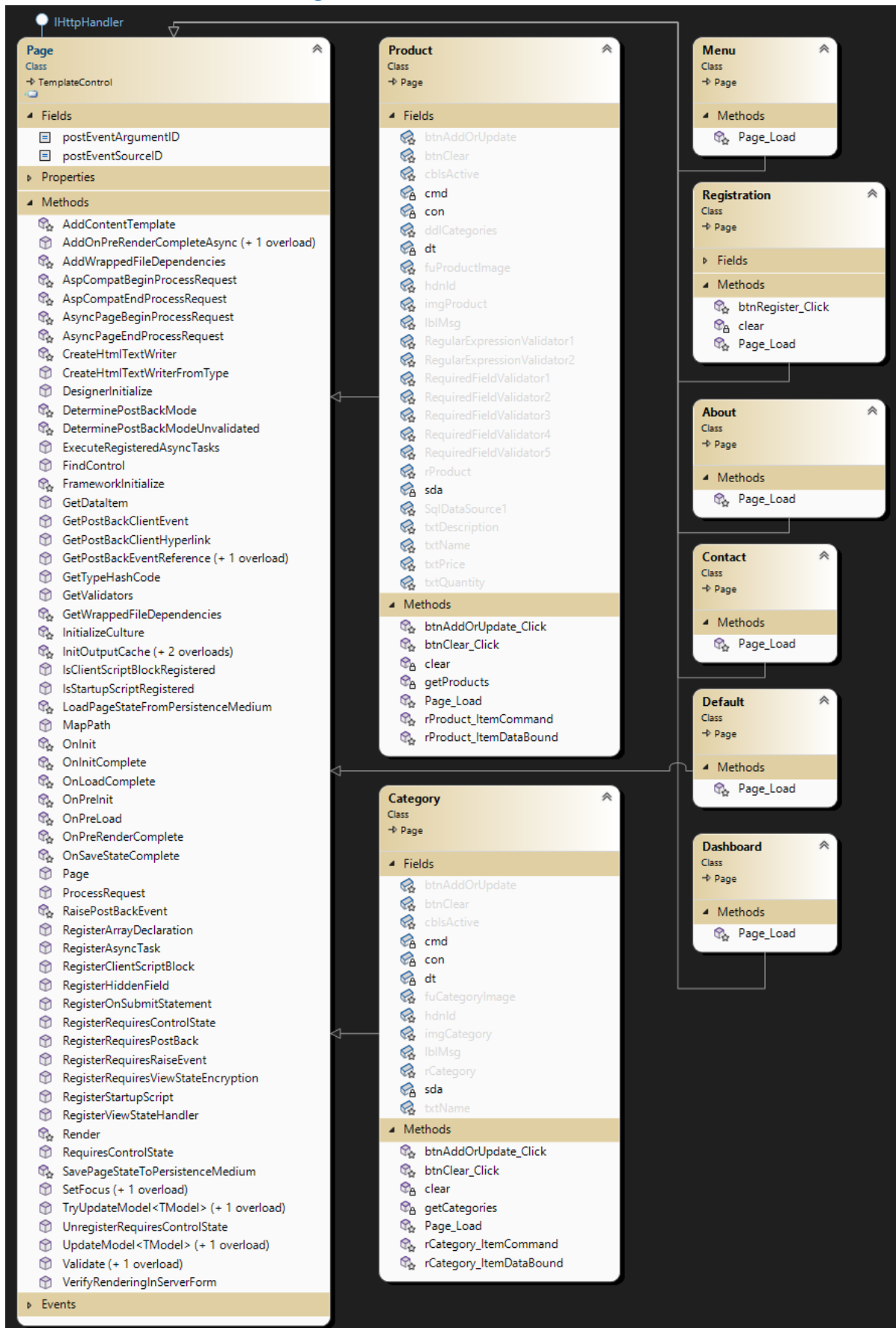


Figure 7. List of classes used in the project.



Figure 8. List of classes used in the project.

2.3. Implementation

2.3.1. Main module of the app (ASP.NET)

Class Registration

A class allowing for the registration of a new user, or update information of currently registered user, includes methods for collecting and validating user input, creating a new user account, and storing the account information in a database.

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;

namespace FitsOnYou.User
{
    2 references
    public partial class Registration : System.Web.UI.Page
    {
        SqlConnection con;
        SqlCommand cmd;
        SqlDataAdapter sda;
        DataTable dt;

        0 references
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        0 references
        protected void btnRegister_Click(object sender, EventArgs e)
        {
            string actionName = string.Empty, imagePath = string.Empty, fileExtension = string.Empty;
            bool isValidToExecute = false;
            int userId = Convert.ToInt32(Request.QueryString["id"]);
            con = new SqlConnection(Connection.GetConnectionString());
            cmd = new SqlCommand("User_Crud", con);
            cmd.Parameters.AddWithValue("@Action", userId == 0 ? "INSERT" : "UPDATE");
            cmd.Parameters.AddWithValue("@UserId", userId);
            cmd.Parameters.AddWithValue("@Name", txtName.Text.Trim());
            cmd.Parameters.AddWithValue("@Username", txtUsername.Text.Trim());
            cmd.Parameters.AddWithValue("@Mobile", txtMobile.Text.Trim());
            cmd.Parameters.AddWithValue("@Email", txtEmail.Text.Trim());
            cmd.Parameters.AddWithValue("@Address", txtAddress.Text.Trim());
            cmd.Parameters.AddWithValue("@PostCode", txtPostCode.Text.Trim());
            cmd.Parameters.AddWithValue("@Password", txtPassword.Text.Trim());
            if (fuUserImage.HasFile)
            {
                if (Utils.IsValidExtension(fuUserImage.FileName))
                {
                    Guid obj = Guid.NewGuid();
                    fileExtension = Path.GetExtension(fuUserImage.FileName);
                    imagePath = "Images/User/" + obj.ToString() + fileExtension;
                    fuUserImage.PostedFile.SaveAs(Server.MapPath("~/Images/User/")
                        + obj.ToString() + fileExtension);
                    cmd.Parameters.AddWithValue("@ImageUrl", imagePath);
                    isValidToExecute = true;
                }
                else
                {
                    lblMsg.Visible = true;
                    lblMsg.Text = "Please select .jpg, .jpeg or .png image";
                    lblMsg.CssClass = "alert alert-danger";
                    isValidToExecute = false;
                }
            }
        }
    }
}

```

Figure 9. Class Registration for UI Registration in User Section

This is one of the interesting examples of how ASP.NET works. Here in the Registration.aspx.cs file we see some C# code containing, for example, the Registration class. However, it is an engine, only a so-called back-end front-end for Registration Page for new user registration. The frontend, in this case, contained in the Registration.aspx file, closely cooperates with the previously mentioned engine and is responsible only for what is displayed in the browser.

Here is the code responsible for designing the Registration Page that the user sees:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/User/User.Master" AutoEventWireup="true" CodeBehind="Registration.aspx.cs"
Inherits="FitsOnYou.User.Registration" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
  <script>
    /*For disappearing alert message*/
    window.onload = function () {
      var second = 5;
      setTimeout(function () {
        document.getElementById("<%=lblMsg.ClientID %>").style.display = "none";
      }, seconds * 1000);
    }
  </script>
  <script>
    function ImagePreview(input) {
      if (input.files && input.files[0]) {
        var reader = new FileReader();
        reader.onload = function (e) {
          $('#<%=imgUser.ClientID %>').prop('src', e.target.result)
            .width(200)
            .height(200);
        };
        reader.readAsDataURL(input.files[0]);
      }
    }
  </script>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <section class="book_section layout_padding">
    <div class="container">
      <div class="heading_container">
        <div class="align-self-end">
          <asp:Label ID="lblMsg" runat="server" Visible="false"></asp:Label>
        </div>
        <asp:Label ID="lblHeaderMsg" runat="server" Text="<h2>User Registration</h2>"></asp:Label>
      </div>
      <div class="row">
        <div class="col-md-6">
          <div class="form_container">
            <div>
              <asp:RequiredFieldValidator ID="rfvName" runat="server" ErrorMessage="Name is required"
                ControlToValidate="txtName"
                ForeColor="Red" Display="Dynamic" SetFocusOnError="true"></asp:RequiredFieldValidator>
              <asp:RegularExpressionValidator ID="revName" runat="server" ErrorMessage="Name must be in characters only"
                ForeColor="Red" Display="Dynamic" SetFocusOnError="true" ValidationExpression="^[a-zA-Z\s]+$"
                ControlToValidate="txtName"></asp:RegularExpressionValidator>
              <asp:TextBox ID="txtName" runat="server" CssClass="form-control" placeholder="Enter Full Name"
                ToolTip="Full Name"></asp:TextBox>
            </div>
            <div>
              <asp:RequiredFieldValidator ID="rfvUsername" runat="server" ErrorMessage="Username is required"
                ControlToValidate="txtUsername" ForeColor="Red" Display="Dynamic"
                SetFocusOnError="true"></asp:RequiredFieldValidator>
              <asp:TextBox ID="txtUsername" runat="server" CssClass="form-control" placeholder="Enter Username"
                ToolTip="Username"></asp:TextBox>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>
</asp:Content>
```

Figure 10. The part of the code related to the Registration Page

It would seem that you can find some elements in common with html, if only because of the similar use of, for example <div>, but only apparently and it is certainly not html.

To visualize the issue, I present the structure of files with marked files related to registration functionality.

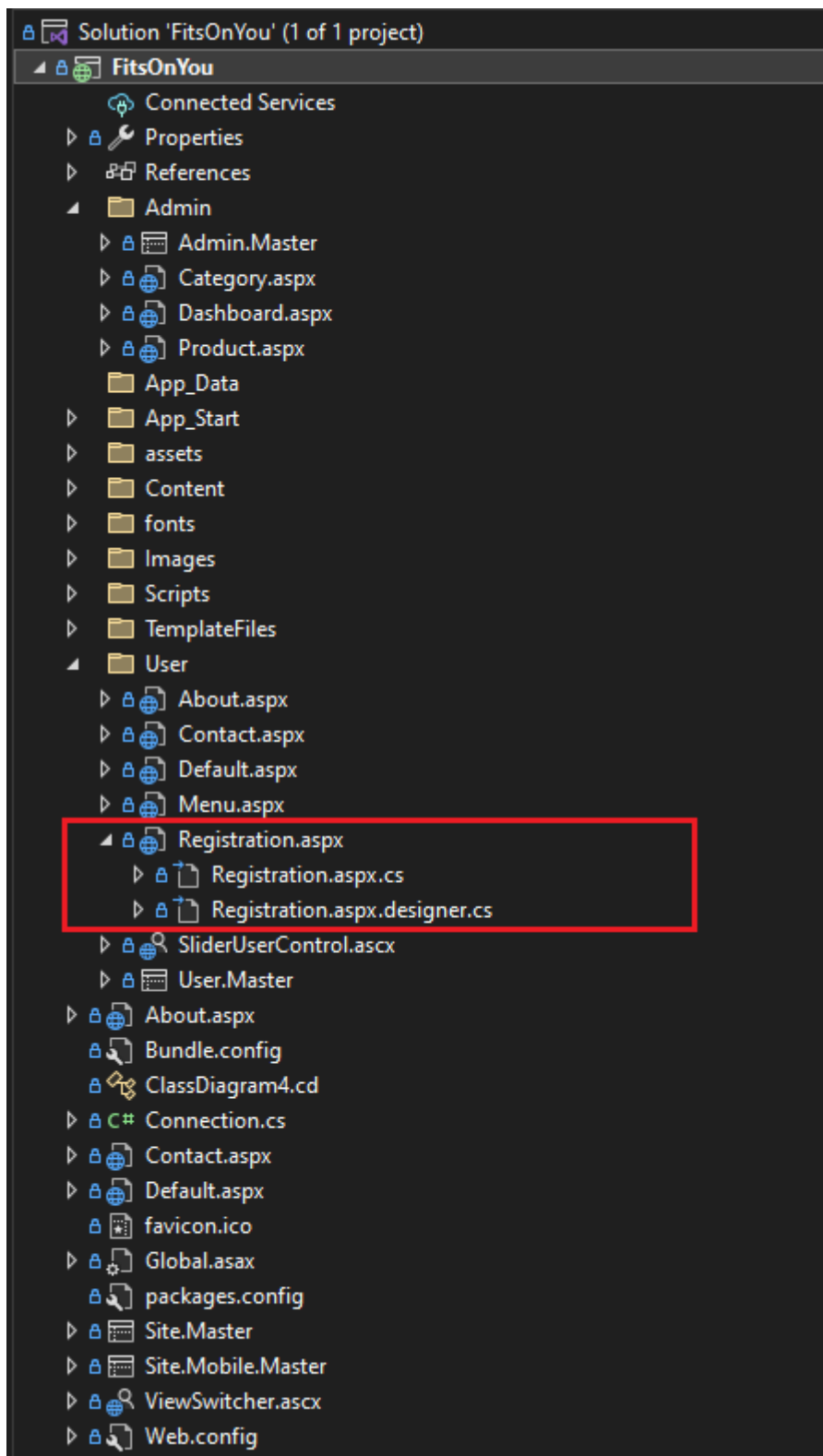


Figure 11. Example file structure for my project in ASP.NET with registration highlighting

Within the class Registration class, I used validation as a safeguard for entering appropriate types of entries in selected fields, e.g., telephone number or e-mail. When creating a new user, the e-mail is checked in the database for uniqueness and whether it is not already used by another user.

```
if (isValidToExecute)
{
    cmd.CommandType = CommandType.StoredProcedure;
    try
    {
        con.Open();
        cmd.ExecuteNonQuery();
        actionName = userId == 0 ?
            "registration is successful! <b><a href='Login.aspx'>Click here</a></b> to do login" :
            "details updated successful! <b><a href='Profile.aspx'>Can check here</a></b>";
        lblMsg.Visible = true;
        lblMsg.Text = "<b> " + txtUsername.Text.Trim() + "</b> " + actionName;
        lblMsg.CssClass = "alert alert-success";
        if (userId != 0)
        {
            Response.AddHeader("REFRESH", "1;URL=Profile.aspx");
        }
        clear();
    }
    catch (SqlException ex)
    {
        if (ex.Message.Contains("Violation of UNIQUE KEY constraint"))
        {
            lblMsg.Visible = true;
            lblMsg.Text = "<b> " + txtUsername.Text.Trim() + "</b> username already exists, try new one...!";
            lblMsg.CssClass = "alert alert-danger";
        }
    }
    catch (Exception ex)
    {
        lblMsg.Visible = true;
        lblMsg.Text = "Error - " + ex.Message;
        lblMsg.CssClass = "alert alert-danger";
    }
    finally
    {
        con.Close();
    }
}
```

Figure 12. Validation of data entered by the user during registration

Class Category

Here, in the next example, I present part of the code for the Category class in the administration section. It is an example of how I connected the functionality for the product categories with the Category table in the database structure.

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Web.UI.WebControls;

namespace FitsOnYou.Admin
{
    2 references
    public partial class Category : System.Web.UI.Page
    {
        SqlConnection con;
        SqlCommand cmd;
        SqlDataAdapter sda;
        DataTable dt;

        0 references
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                Session["breadCrum"] = "Category";
                getCategories();
            }
            lblMsg.Visible = false;
        }

        0 references
        protected void btnAddOrUpdate_Click(object sender, EventArgs e)
        {
            string actionName = string.Empty, imagePath = string.Empty, fileExtension = string.Empty;
            bool isValidToExecute = false;
            int categoryId = Convert.ToInt32(hdnId.Value);
            con = new SqlConnection(Connection.GetConnectionString());
            cmd = new SqlCommand("Category_Crud", con);
            cmd.Parameters.AddWithValue("@Action", categoryId == 0 ? "INSERT" : "UPDATE");
            cmd.Parameters.AddWithValue("@CategoryId", categoryId);
            cmd.Parameters.AddWithValue("@Name", txtName.Text.Trim());
            cmd.Parameters.AddWithValue("@IsActive", cbIsActive.Checked);
        }
    }
}
```

Figure 13. Part of the code in the Category class responsible for connecting the functionality for the product category with the database.

Here is part of the code responsible for adding a new category to the database.

```
3 references
private void getCategories()
{
    con = new SqlConnection(Connection.GetConnectionString());
    cmd = new SqlCommand("Category_Crud", con);
    cmd.Parameters.AddWithValue("@Action", "SELECT");
    cmd.CommandType = CommandType.StoredProcedure;
    sda = new SqlDataAdapter(cmd);
    dt = new DataTable();
    sda.Fill(dt);
    rCategory.DataSource = dt;
    rCategory.DataBind();
}
}
```

Figure 14. Part of the code responsible for adding a new category to database

Here is part of the code responsible for edit and delete a category in database within Category class in the administration section.

```
protected void rCategory_ItemCommand(object source, RepeaterCommandEventArgs e)
{
    lblMsg.Visible = false;
    if (e.CommandName == "edit")
    {
        con = new SqlConnection(Connection.GetConnectionString());
        cmd = new SqlCommand("Category_Crud", con);
        cmd.Parameters.AddWithValue("@Action", "GETBYID");
        cmd.Parameters.AddWithValue("@CategoryId", e.CommandArgument);
        cmd.CommandType = CommandType.StoredProcedure;
        sda = new SqlDataAdapter(cmd);
        dt = new DataTable();
        sda.Fill(dt);
        txtName.Text = dt.Rows[0]["Name"].ToString();
        cbIsActive.Checked = Convert.ToBoolean(dt.Rows[0]["IsActive"]);
        imgCategory.ImageUrl = string.IsNullOrEmpty(dt.Rows[0]["ImageUrl"].ToString()) ?
            "../Images/No_image.png" : "../" + dt.Rows[0]["ImageUrl"].ToString();
        imgCategory.Height = 200;
        imgCategory.Width = 200;
        hdnId.Value = dt.Rows[0]["CategoryId"].ToString();
        btnAddOrUpdate.Text = "Update";
        LinkButton btn = e.Item.FindControl("lnkEdit") as LinkButton;
        btn.CssClass = "badge badge-warning";
    }
    else if (e.CommandName == "delete")
    {
        con = new SqlConnection(Connection.GetConnectionString());
        cmd = new SqlCommand("Category_Crud", con);
        cmd.Parameters.AddWithValue("@Action", "DELETE");
        cmd.Parameters.AddWithValue("@CategoryId", e.CommandArgument);
        cmd.CommandType = CommandType.StoredProcedure;
        try
        {
            con.Open();
            cmd.ExecuteNonQuery();
            lblMsg.Visible = true;
            lblMsg.Text = "Category deleted successfully!";
            lblMsg.CssClass = "alert alert-success";
            getCategories();
        }
        catch (Exception ex)
        {
            lblMsg.Visible = true;
            lblMsg.Text = "Error - " + ex.Message;
            lblMsg.CssClass = "alert alert-danger";
        }
        finally
        {
            con.Close();
        }
    }
}
```

Figure 15. Part of the code responsible for edit and delete a category in database

In order for all CRUD functionalities for the above Category to work properly, I had to properly configure Stored Procedures for the database using specially created queries, which were then invoked in Microsoft SQL Server Management.

```

ALTER PROCEDURE [dbo].[Category_Crud]
-- Add the parameters for the stored procedure here
@Action VARCHAR(10),
@CategoryId INT = NULL,
@Name VARCHAR(100) = NULL,
@IsActive BIT = false,
@ImageUrl VARCHAR(MAX) = NULL
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

--SELECT
IF @Action = 'SELECT'
BEGIN
SELECT * FROM dbo.Categories ORDER BY CreatedDate DESC
END

--INSERT
IF @Action = 'INSERT'
BEGIN
INSERT INTO dbo.Categories(Name, ImageUrl, IsActive, CreatedDate)
VALUES (@Name, @ImageUrl, @IsActive, GETDATE())
END

--UPDATE
IF @Action = 'UPDATE'
BEGIN
DECLARE @UPDATE_IMAGE VARCHAR(20)
SELECT @UPDATE_IMAGE = (CASE WHEN @ImageUrl IS NULL THEN 'NO' ELSE 'YES' END)
IF @UPDATE_IMAGE = 'NO'
BEGIN
UPDATE dbo.Categories
SET Name = @Name, IsActive = @IsActive
WHERE CategoryId = @CategoryId
END
ELSE
BEGIN
UPDATE dbo.Categories
SET Name = @Name, ImageUrl = @ImageUrl, IsActive = @IsActive
WHERE CategoryId = @CategoryId
END
END

--DELETE
IF @Action = 'DELETE'
BEGIN
DELETE FROM dbo.Categories WHERE CategoryId = @CategoryId
END

--GETBYID
IF @Action = 'GETBYID'
BEGIN
SELECT * FROM dbo.Categories WHERE CategoryId = @CategoryId
END
END

```

Figure 16. Stored Procedures for Category table

The connection to the database, created in Server Management Studio Management Studio 19 and running in Microsoft SQL Server running in the background, is made via the appropriate script in the Web.config configuration file.

```
<appSettings>
  <add key="username" value="Admin"/>
  <add key="password" value="123456"/>
  <add key="ValidationSettings:UnobtrusiveValidationMode" value="None"/>
</appSettings>

<connectionStrings>
  <add name="cs" connectionString="Data Source=.\SQLEXPRESS;
  Initial Catalog=FitsOnYouDB;
  Integrated Security=True;
  MultipleActiveResultSets=true;"
  providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Figure 17. The part of script for connecting a database

2.3.2. Video-chat module of app (NodeJS, React)

The application consists of two modules: backend and client.

2.3.2.1. Backend of video-chat application

Index.js

Initiates run of the chat application as a server side. Backend of this web application consists of a simple NodeJS server which runs on port 5000.

```
JS index.js > app.get('/') callback
1  const app = require("express")();
2  const server = require("http").createServer(app);
3  const cors = require("cors");
4
5  const io = require("socket.io")(server, {
6    cors: {
7      origin: "*",
8      methods: [ "GET", "POST" ]
9    }
10 });
11
12 app.use(cors());
13
14 const PORT = process.env.PORT || 5000;
15
16 app.get('/', (req, res) => {
17   res.send('Server is running...');
18 });
19
```

Figure 18. Index.js file from backend of video-chat

```

20  ✓ io.on("connection", (socket) => {
21      socket.emit("me", socket.id);
22
23  ✓      socket.on("disconnect", () => {
24          socket.broadcast.emit("callEnded")
25      });
26
27  ✓      socket.on("callUser", ({ userToCall, signalData, from, name }) => {
28          io.to(userToCall).emit("callUser", { signal: signalData, from, name });
29      });
30
31  ✓      socket.on("answerCall", (data) => {
32          io.to(data.to).emit("callAccepted", data.signal)
33      });
34  });
35
36  server.listen(PORT, () => console.log(`Server is running on port ${PORT}`));

```

Figure 19. Continue of Index.js file from backend of video-chat

The first step is to initialize the socket.io server library, so we use the **express** module, and then, for the socket server instance, we use the **HTTP.createServer()** method.

```

const app = require("express")();
const server = require("http").createServer(app);

```

Now we have the ability to emit and listen to events between the server and the client. The emission of the **"me"** message is our identifier. It is passed via **socket.id**.

```

io.on("connection", (socket) => {
    socket.emit("me", socket.id);
});

```

The next step is to issue the **callEnded** message, which is responsible for ending the connection in the connection handling module.

```

socket.on("disconnect", () => {
    socket.broadcast.emit("callEnded")
});

```

The **callUser** handler is responsible for the call initiation function, which contains parameters such as **userToCall**, **signalData**, **from**, **name**.

```

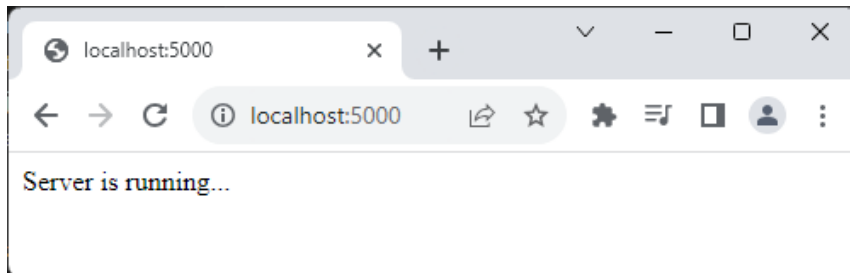
socket.on("callUser", ({ userToCall, signalData, from, name }) => {
    io.to(userToCall).emit("callUser", { signal: signalData, from, name });
});

```

In `answerCall` we get feedback from the client.

```
socket.on("answerCall", (data) => {  
  io.to(data.to).emit("callAccepted", data.signal)  
});
```

Below, using a browser, we can observe a running server on the localhost on port 5000.



I managed to deploy backend (server side) to the Heroku platform, and it is available at the link below, which is used in the main part of my application:

<https://fitsonyou-videochat-fa37f7f15fbf.herokuapp.com/>

To demonstrate how works socket.io I use a localhost on port 5000 where our server is running. In this case the client side of the video chat application communicates with the server through this port. In my main application I use a link to the currently deployed by me on Heroku server version side of video-chat app (as you can see in the commented line below).

```
client > src > JS Context.js > [e] ContextProvider  
7   const socket = io('http://localhost:5000');  
8   //const socket = io('https://fitsonyou-videochat-fa37f7f15fbf.herokuapp.com/');  
9
```

Figure 20. A fragment of a `Context.js` file from frontend of video-chat

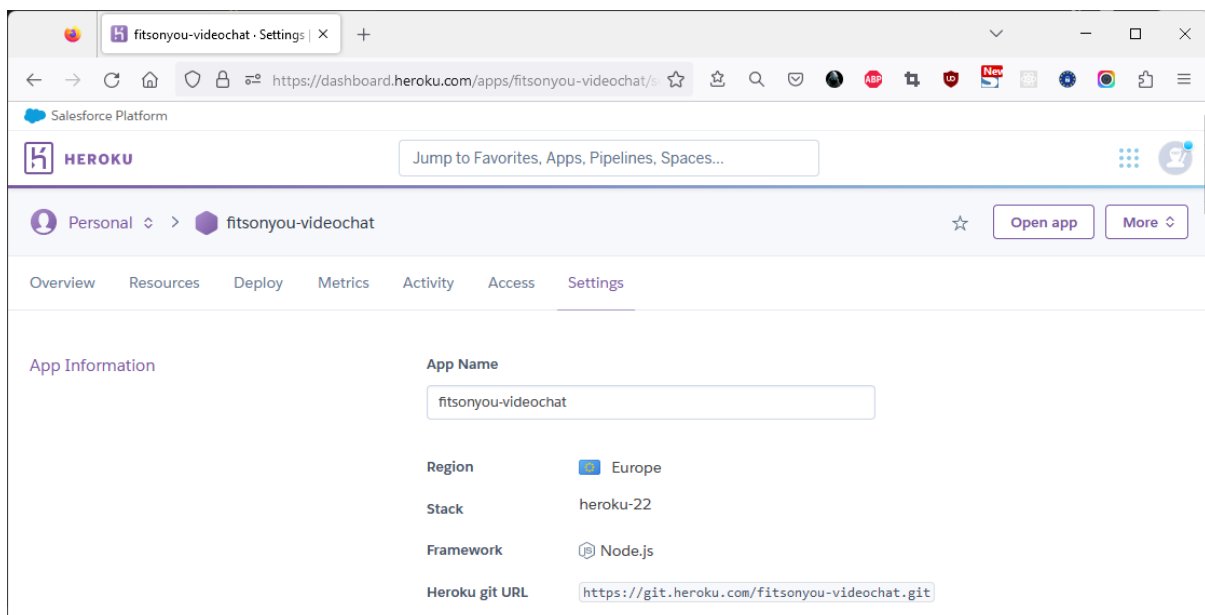


Figure 21. The view of Heroku server on which backend video-chat was deployed

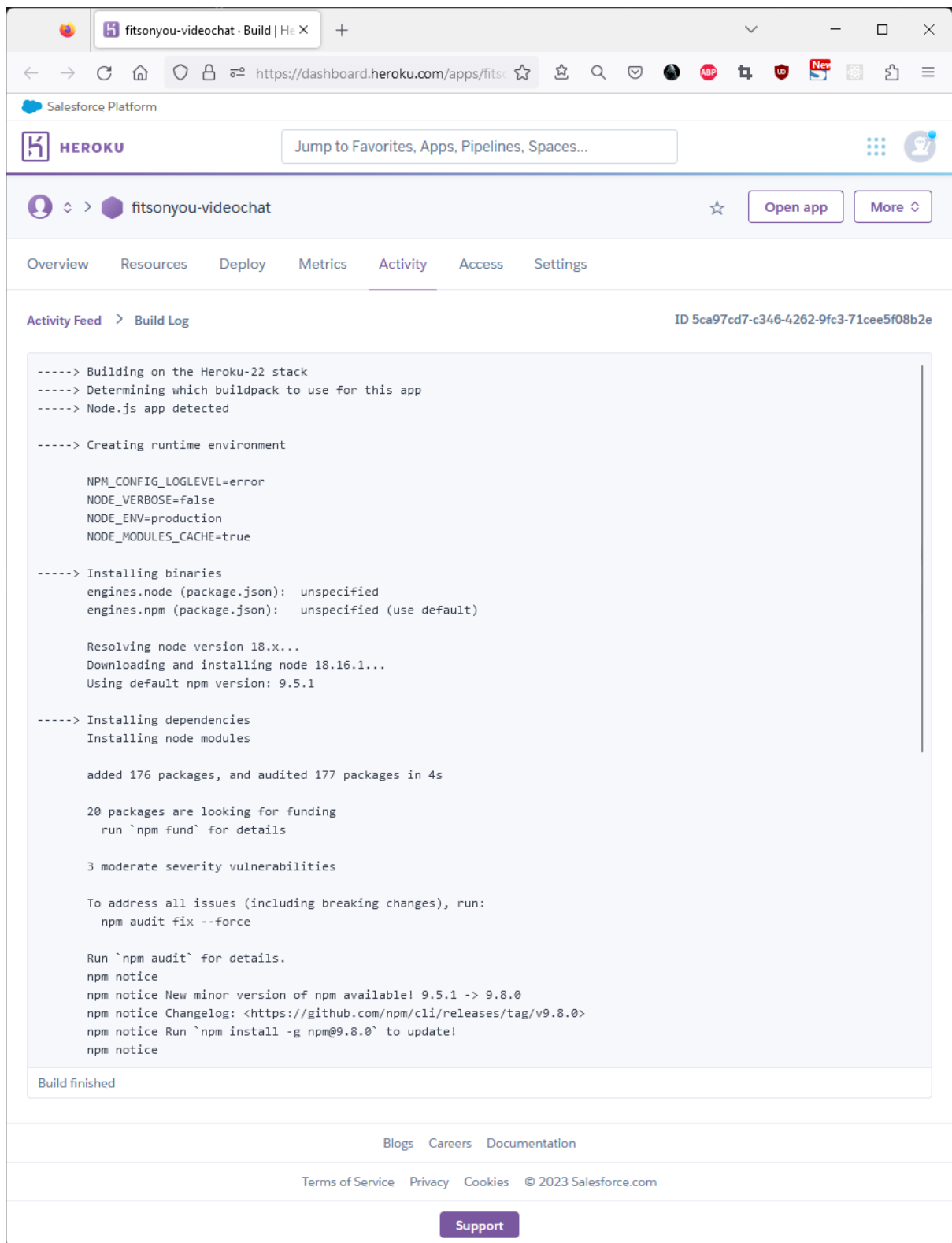


Figure 22. The view of deploying logs at server Heroku

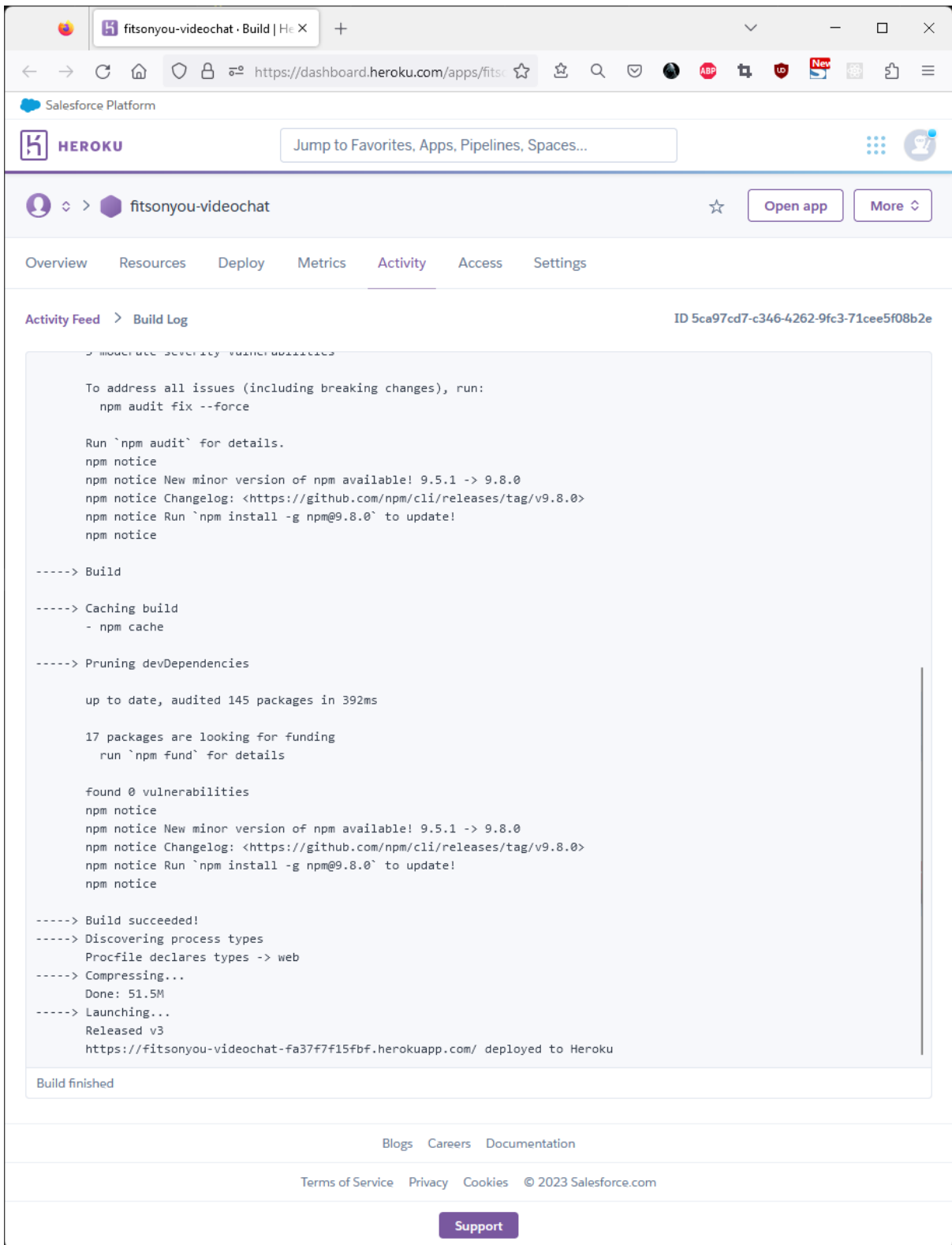


Figure 23. Continue of view of deploying logs on the server Heroku

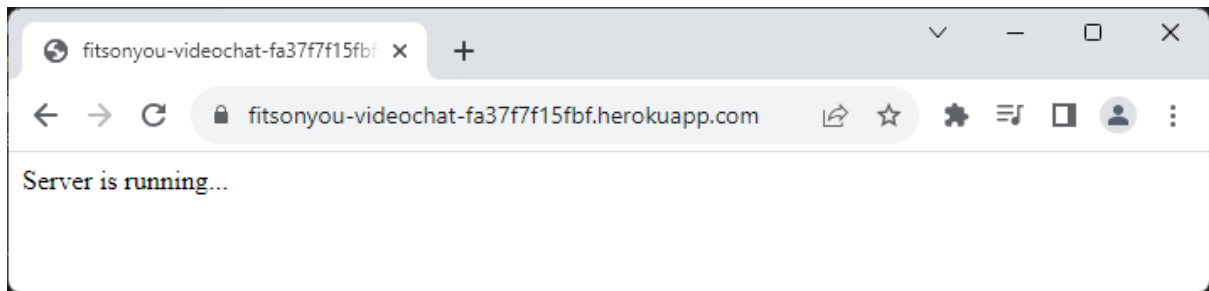


Figure 24. The running server from a browser view

2.3.2.2. Frontend of chat application (client)

The client module is responsible for the operation of video-chat. I used the REACT technology.

App.js

```
JS App.js x
client > src > JS App.js > ...
 1  import React from 'react';
 2  import { Typography, AppBar } from '@material-ui/core';
 3  import { makeStyles } from '@material-ui/core/styles';
 4
 5  import VideoPlayer from './components/VideoPlayer';
 6  import Sidebar from './components/Sidebar';
 7  import Notifications from './components/Notifications';
 8
 9  const useStyles = makeStyles((theme) => ({
10    appBar: {
11      borderRadius: 15,
12      margin: '30px 100px',
13      display: 'flex',
14      flexDirection: 'row',
15      justifyContent: 'center',
16      alignItems: 'center',
17      width: '600px',
18      border: '2px solid black',
19
20      [theme.breakpoints.down('xs')]: {
21        width: '90%',
22      },
23    },
24    image: {
25      marginLeft: '15px',
26    },

```

Figure 25.App.js file in fronted of video-chat

```

27   wrapper: {
28     display: 'flex',
29     flexDirection: 'column',
30     alignItems: 'center',
31     width: '100%',
32   },
33   });
34
35   const App = () => {
36     const classes = useStyles();
37
38     return (
39       <div className={classes.wrapper}>
40         <AppBar className={classes.appBar} position="static" color="inherit">
41           <Typography variant="h2" align="center">Video Chat</Typography>
42         </AppBar>
43         <VideoPlayer />
44         <Sidebar>
45           <Notifications />
46         </Sidebar>
47       </div>
48     );
49   };
50
51   export default App;

```

Figure 26. Continue of App.js file in fronted of video-chat

Index.js

```

JS index.js  X
client > src > JS index.js
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3
4   import App from './App';
5   import { ContextProvider } from './Context';
6
7   import './styles.css';
8
9   ReactDOM.render(
10    <ContextProvider>
11      <App />
12    </ContextProvider>,
13    document.getElementById('root'),
14  );
15

```

Figure 27. Index.js file in fronted of video-chat

Context.js

In the code presented below, the so-called back-end of the frontend is included. For the purposes of demonstrating how the code works, the socket.io application broadcasts using localhost on port 5000, where our server is running. In the main application at this point (as you can see in the commented line) I use a link to the currently deployed version by me on the Heroku server.

```
client > src > JS Context.js > ContextProvider
1  import React, { createContext, useState, useRef, useEffect } from 'react';
2  import { io } from 'socket.io-client';
3  import Peer from 'simple-peer';
4
5  const SocketContext = createContext();
6
7  const socket = io('http://localhost:5000');
8  //const socket = io('https://fitsonyou-videochat-fa37f7f15fbf.herokuapp.com/');
9
10 const ContextProvider = ({ children }) => {
11   const [callAccepted, setCallAccepted] = useState(false);
12   const [callEnded, setCallEnded] = useState(false);
13   const [stream, setStream] = useState();
14   const [name, setName] = useState('');
15   const [call, setCall] = useState({});
16   const [me, setMe] = useState('');
17
18   const myVideo = useRef();
19   const userVideo = useRef();
20   const connectionRef = useRef();
21
22   useEffect(() => {
23     navigator.mediaDevices.getUserMedia({ video: true, audio: true })
24       .then((currentStream) => {
25         setStream(currentStream);
26
27         myVideo.current.srcObject = currentStream;
28       });
29
30     socket.on('me', (id) => setMe(id));
31
32     socket.on('callUser', ({ from, name: callerName, signal }) => {
33       setCall({ isReceivingCall: true, from, name: callerName, signal });
34     });
35   }, []);
36
37   const answerCall = () => {
38     setCallAccepted(true);
39
40     const peer = new Peer({ initiator: false, trickle: false, stream });
41
42     peer.on('signal', (data) => {
43       socket.emit('answerCall', { signal: data, to: call.from });
44     });
```

Figure 28. Context.js file in fronted of video-chat

```

client > src > JS Context.js > ContextProvider
44     });
45
46     peer.on('stream', (currentStream) => {
47         |   userVideo.current.srcObject = currentStream;
48     });
49
50     peer.signal(call.signal);
51
52     connectionRef.current = peer;
53 };
54
55 const callUser = (id) => {
56     |   const peer = new Peer({ initiator: true, trickle: false, stream });
57
58     |   peer.on('signal', (data) => {
59         |       |   socket.emit('callUser', { userToCall: id, signalData: data, from: me, name });
60     });
61
62     |   peer.on('stream', (currentStream) => {
63         |       |   userVideo.current.srcObject = currentStream;
64     });
65
66     |   socket.on('callAccepted', (signal) => {
67         |       |   setCallAccepted(true);
68
69         |       |   peer.signal(signal);
70     });
71
72     |   connectionRef.current = peer;
73 };
74
75 const leaveCall = () => {
76     |   setCallEnded(true);
77
78     |   connectionRef.current.destroy();
79
80     |   window.location.reload();
81 };
82
83 return (
84     |   <SocketContext.Provider value={{
85         |       call,
86         |       callAccepted,
87         |       myVideo,

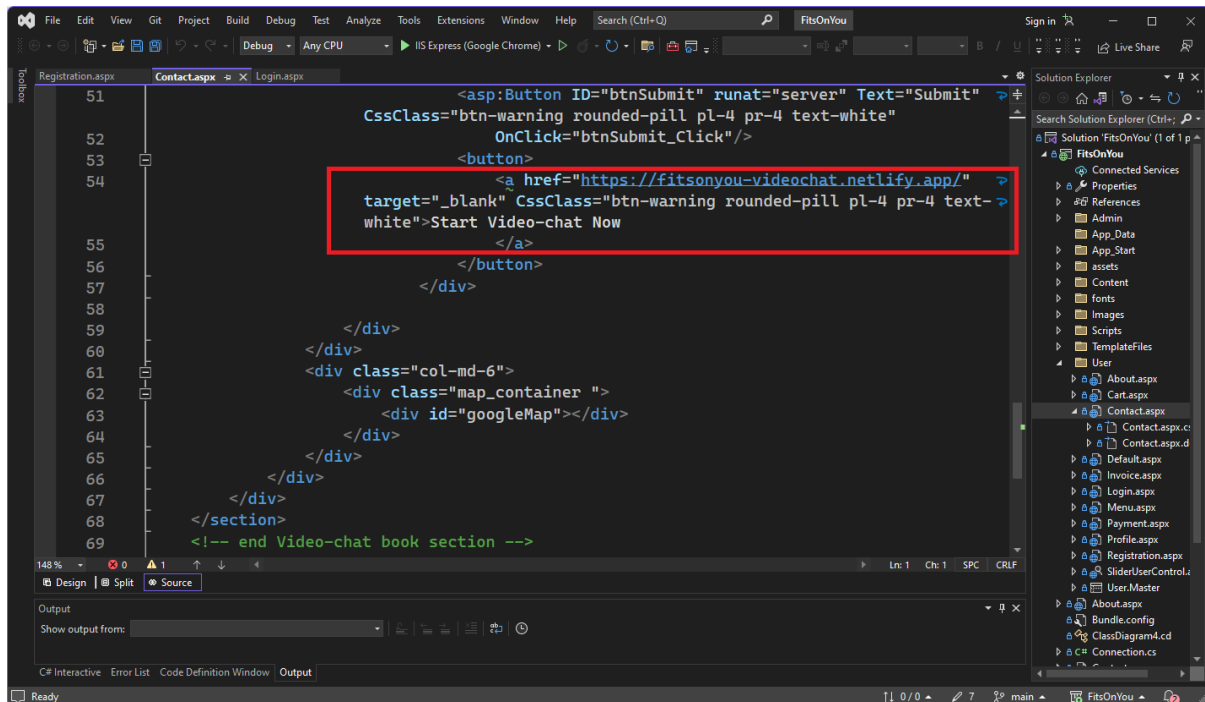
```

Figure 29. Continue of Context.js file in fronted of video-chat

```
JS Context.js X
client > src > JS Context.js > ...
54
55 const callUser = (id) => {
56   const peer = new Peer({ initiator: true, trickle: false, stream });
57
58   peer.on('signal', (data) => {
59     socket.emit('callUser', { userToCall: id, signalData: data, from: me, name });
60   });
61
62   peer.on('stream', (currentStream) => {
63     userVideo.current.srcObject = currentStream;
64   });
65
66   socket.on('callAccepted', (signal) => {
67     setCallAccepted(true);
68
69     peer.signal(signal);
70   });
71
72   connectionRef.current = peer;
73 };
74
75 const leaveCall = () => {
76   setCallEnded(true);
77
78   connectionRef.current.destroy();
79
80   window.location.reload();
81 };
82
83 return (
84   <SocketContext.Provider value={{
85     call,
86     callAccepted,
87     myVideo,
88     userVideo,
89     stream,
90     name,
91     setName,
92     callEnded,
93     me,
94     callUser,
95     leaveCall,
96     answerCall,
97   }}
98   >
99   {children}
100 </SocketContext.Provider>
101 );
102 };
103
104 export { ContextProvider, SocketContext };
105
```

Figure 30. Continue of Context.js file in fronted of video-chat

Below is a code fragment in which I use a link to the part of the application responsible for the video-chat front-end deployed on the Netlify server by me.



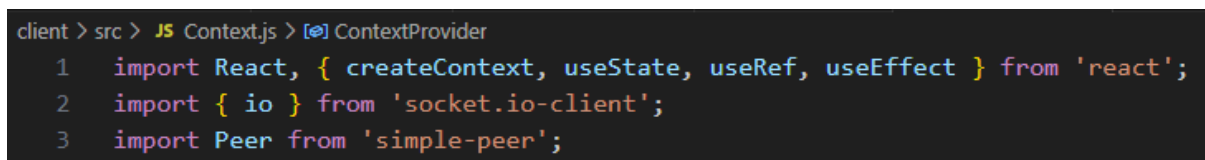
The screenshot shows a Visual Studio Code editor with a file named `Contact.aspx` open. The code is in ASP.NET Razor syntax. A red box highlights the following code fragment:

```
<a href="https://fitsonyou-videochat.netlify.app/"
target="_blank" CssClass="btn-warning rounded-pill pl-4 pr-4 text-
white">Start Video-chat Now
</a>
```

The rest of the code in the file includes a submit button, a map container, and a section comment. The Solution Explorer on the right shows the project structure for 'FitsOnYou'.

Figure 31. A fragment of the code from main application with link to video-chat deployed on Netlify

All socket.io functionality is contained in the Context.js file to keep it in the react context. Below we can see the necessary components imported from React, i.e., socket.io-client and simple-peer.



```
client > src > JS Context.js > [⌘] ContextProvider
1 import React, { createContext, useState, useRef, useEffect } from 'react';
2 import { io } from 'socket.io-client';
3 import Peer from 'simple-peer';
```

Figure 32. Imports in Context.js file in frontend of video-chat app

The three React components used in the app

- **VideoPlayer.js**

```
VideoPlayer.jsx X
client > src > components > VideoPlayer.jsx > ...
1 import React, { useContext } from 'react';
2 import { Grid, Typography, Paper, makeStyles } from '@material-ui/core';
3
4 import { SocketContext } from '../Context';
5
6 const useStyles = makeStyles((theme) => ({
7   video: {
8     width: '550px',
9     [theme.breakpoints.down('xs')]: {
10      width: '400px',
11    },
12  },
13  gridContainer: {
14    justifyContent: 'center',
15    [theme.breakpoints.down('xs')]: {
16      flexDirection: 'column',
17    },
18  },
19  paper: {
20    padding: '10px',
21    border: '2px solid black',
22    margin: '10px',
23  },
24 }));
25
26 const VideoPlayer = () => {
27   const { name, callAccepted, myVideo, userVideo, callEnded, stream, call } = useContext(SocketContext);
28   const classes = useStyles();
29
30   return (
31     <Grid container className={classes.gridContainer}>
32       {stream && (
33         <Paper className={classes.paper}>
34           <Grid item xs={12} md={6}>
35             <Typography variant="h5" gutterBottom>{name || 'Name'}</Typography>
36             <video playsInline muted ref={myVideo} autoPlay className={classes.video} />
37           </Grid>
38         </Paper>
39       )}
40       {callAccepted && !callEnded && (
41         <Paper className={classes.paper}>
42           <Grid item xs={12} md={6}>
43             <Typography variant="h5" gutterBottom>{call.name || 'Name'}</Typography>
44             <video playsInline ref={userVideo} autoPlay className={classes.video} />
45           </Grid>
46         </Paper>
47       )}
48     </Grid>
49   );
50 };
51
52 export default VideoPlayer;
```

Figure 33. The first component – VideoPlayer.js file

- Sidebar.js

```
JS Sidebar.jsx X
client > src > components > JS Sidebar.jsx > [0] Sidebar
1  import React, { useState, useContext } from 'react';
2  import { Button, TextField, Grid, Typography, Container, Paper } from '@material-ui/core'
3  import { CopyToClipboard } from 'react-copy-to-clipboard';
4  import { Assignment, Phone, PhoneDisabled } from '@material-ui/icons';
5  import { makeStyles } from '@material-ui/core/styles';
6
7  import { SocketContext } from '../Context';
8
9  const useStyles = makeStyles((theme) => ({
10   root: {
11     display: 'flex',
12     flexDirection: 'column',
13   },
14   gridContainer: {
15     width: '100%',
16     [theme.breakpoints.down('xs')]: {
17       flexDirection: 'column',
18     },
19   },
20   container: {
21     width: '600px',
22     margin: '35px 0',
23     padding: 0,
24     [theme.breakpoints.down('xs')]: {
25       width: '80%',
26     },
27   },
28   margin: {
29     marginTop: 20,
30   },
31   padding: {
32     padding: 20,
33   },
34   paper: {
35     padding: '10px 20px',
36     border: '2px solid black',
37   },
38 }));
39
40 const Sidebar = ({ children }) => {
41   const { me, callAccepted, name, setName, callEnded, leaveCall, callUser } = useContext
42   (SocketContext);
43   const [idToCall, setIdToCall] = useState('');
44   const classes = useStyles();
45
46   return (
47     <Container className={classes.container}>
48       <Paper elevation={10} className={classes.paper}>
49         <form className={classes.root} noValidate autoComplete="off">
50           <Grid container className={classes.gridContainer}>
51             <Grid item xs={12} md={6} className={classes.padding}>
```

Figure 34. The second component – Sidebar.js file

```

51     <Typography gutterBottom variant="h6">Account Info</Typography>
52     <TextField label="Name" value={name} onChange={(e) => setName(e.target.
53     value)} fullWidth />
54     /* {console.log(me)} */
55     <CopyToClipboard text={me} className={classes.margin}>
56     <Button variant="contained" color="primary" fullWidth startIcon=
57     {<Assignment fontSize="large" />}>
58     Copy Your ID
59     </Button>
60     </CopyToClipboard>
61     </Grid>
62     <Grid item xs={12} md={6} className={classes.padding}>
63     <Typography gutterBottom variant="h6">Make a call</Typography>
64     <TextField label="ID to call" value={idToCall} onChange={(e) => setIdToCall
65     (e.target.value)} fullWidth />
66     {callAccepted && !callEnded ? (
67     <Button variant="contained" color="secondary" startIcon={<PhoneDisabled
68     fontSize="large" />} fullWidth onClick={leaveCall} className={classes.
69     margin}>
70     Hang Up
71     </Button>
72     ) : (
73     <Button variant="contained" color="primary" startIcon={<Phone
74     fontSize="large" />} fullWidth onClick={() => callUser(idToCall)}
75     className={classes.margin}>
76     Call
77     </Button>
78     )}
79     </Grid>
80     </Grid>
81     </form>
82     {children}
83     </Paper>
84     </Container>
85     );
86 };
87
88 export default Sidebar;

```

Figure 35. Continue of the second component – Sidebar.js file

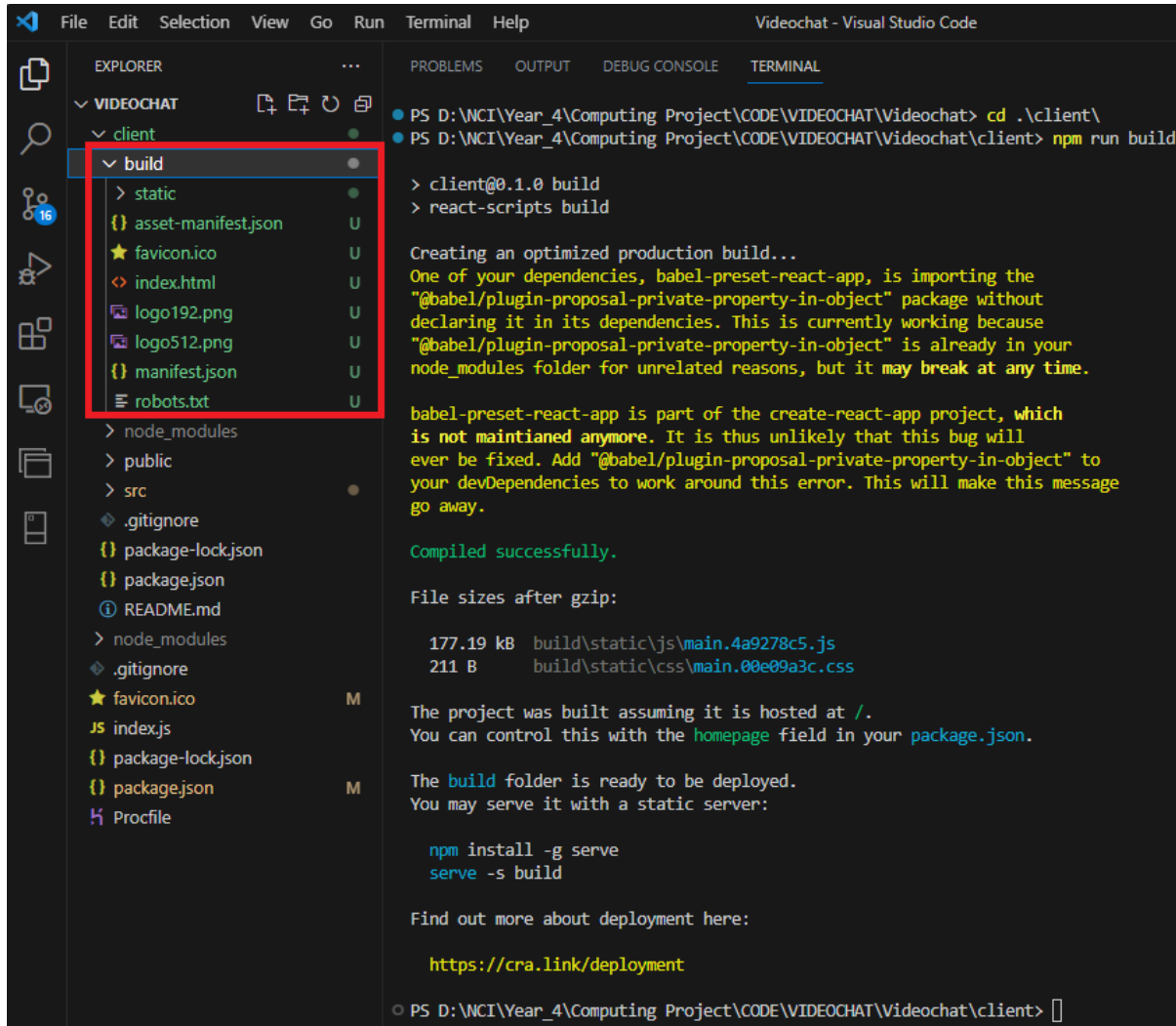
- Notifications.js

```
JS Notifications.jsx X
client > src > components > JS Notifications.jsx > ...
1  import React, { useContext } from 'react';
2  import { Button } from '@material-ui/core';
3
4  import { SocketContext } from '../Context';
5
6  const Notifications = () => {
7    const { answerCall, call, callAccepted } = useContext(SocketContext);
8
9    return (
10     <>
11       {call.isReceivingCall && !callAccepted && (
12         <div style={{ display: 'flex', justifyContent: 'space-around' }}>
13           <h1>{call.name} is calling:</h1>
14           <Button variant="contained" color="primary" onClick={answerCall}>
15             Answer
16           </Button>
17         </div>
18       )}
19     </>
20   );
21 };
22
23 export default Notifications;
24
```

Figure 36. The third component – Notifications.js file

Folder “build”

The last stage of the video-chat application development was to prepare an optimized production version for deploying the front-end to the web. Command `npm run build` bundle all react components and Java Script files into simply html and CSS files ready to use to hosting Netlify platform.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane displays the file structure of the 'VIDEOCHAT' project. The 'client' folder is expanded, and the 'build' folder is highlighted with a red box. Inside 'build', there is a 'static' folder and several files: 'asset-manifest.json', 'favicon.ico', 'index.html', 'logo192.png', 'logo512.png', 'manifest.json', and 'robots.txt'. The main editor area shows the terminal output of the 'npm run build' command. The output indicates that the build was successful and provides instructions for serving the built files.

```
PS D:\NCI\Year_4\Computing Project\CODE\VIDEOCHAT\Videochat> cd .\client\  
PS D:\NCI\Year_4\Computing Project\CODE\VIDEOCHAT\Videochat\client> npm run build  
  
> client@0.1.0 build  
> react-scripts build  
  
Creating an optimized production build...  
One of your dependencies, babel-preset-react-app, is importing the  
"@babel/plugin-proposal-private-property-in-object" package without  
declaring it in its dependencies. This is currently working because  
"@babel/plugin-proposal-private-property-in-object" is already in your  
node_modules folder for unrelated reasons, but it may break at any time.  
  
babel-preset-react-app is part of the create-react-app project, which  
is not maintained anymore. It is thus unlikely that this bug will  
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to  
your devDependencies to work around this error. This will make this message  
go away.  
  
Compiled successfully.  
  
File sizes after gzip:  
  
177.19 kB build\static\js\main.4a9278c5.js  
211 B build\static\css\main.00e09a3c.css  
  
The project was built assuming it is hosted at /.  
You can control this with the homepage field in your package.json.  
  
The build folder is ready to be deployed.  
You may serve it with a static server:  
  
npm install -g serve  
serve -s build  
  
Find out more about deployment here:  
  
https://cra.link/deployment  
  
PS D:\NCI\Year_4\Computing Project\CODE\VIDEOCHAT\Videochat\client>
```

Figure 37. Development the video-chat by build of the code

For developing video-chat I chose the Netlify platform, on which I deployed the frontend of video-chat.

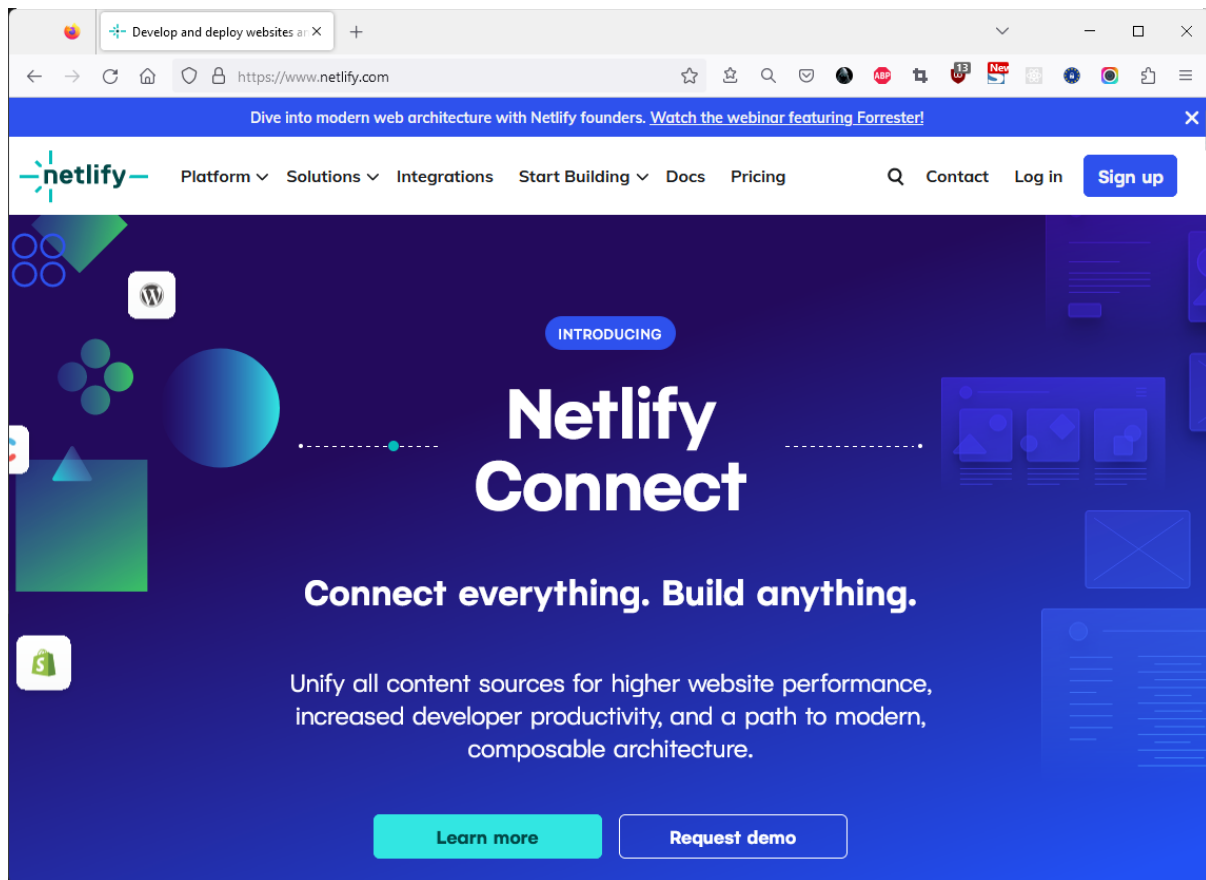


Figure 38. Netlify – web development platform

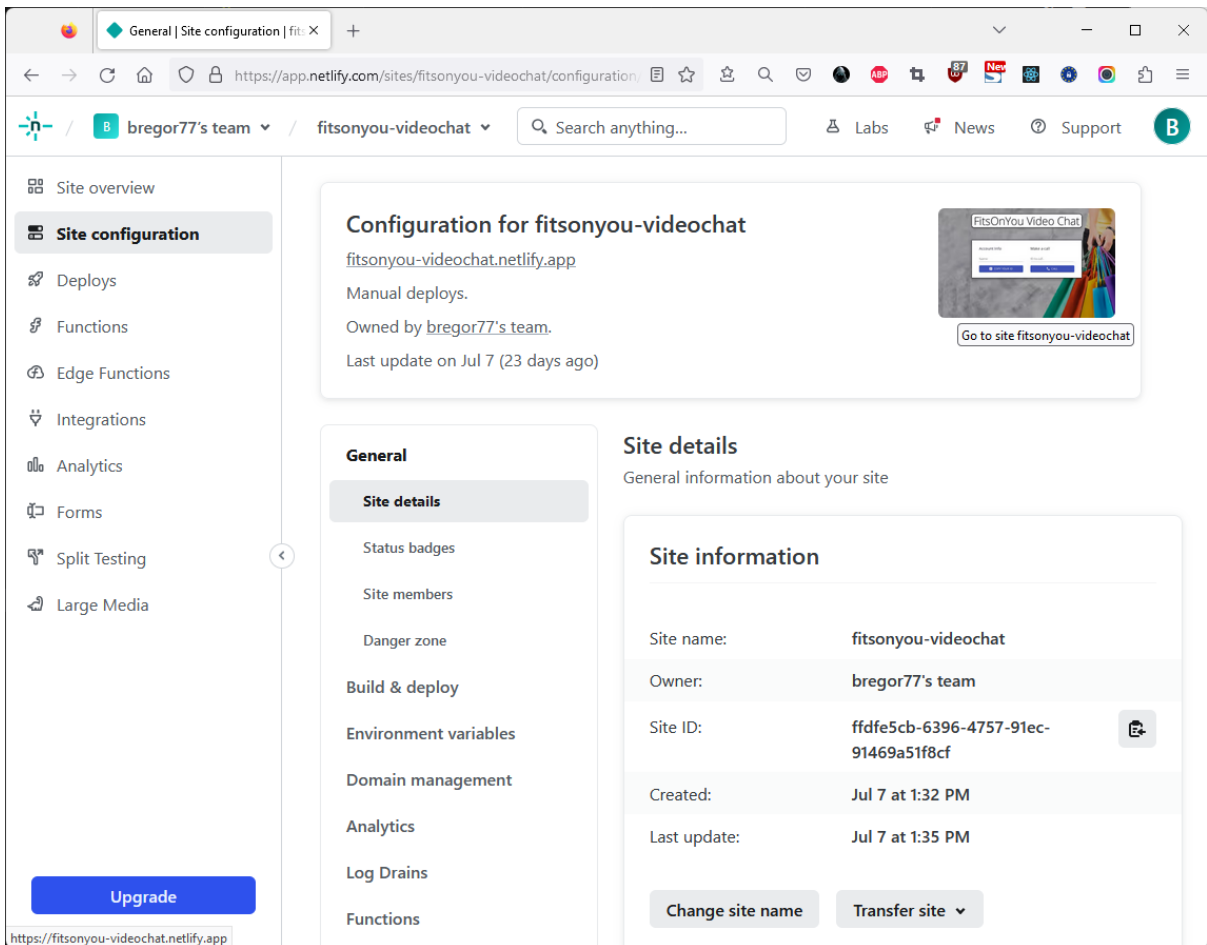


Figure 39. The view of video-chat app already deployed at Netlify

For the full usability of the application, the browser asks the user to enable the use of the camera and microphone.

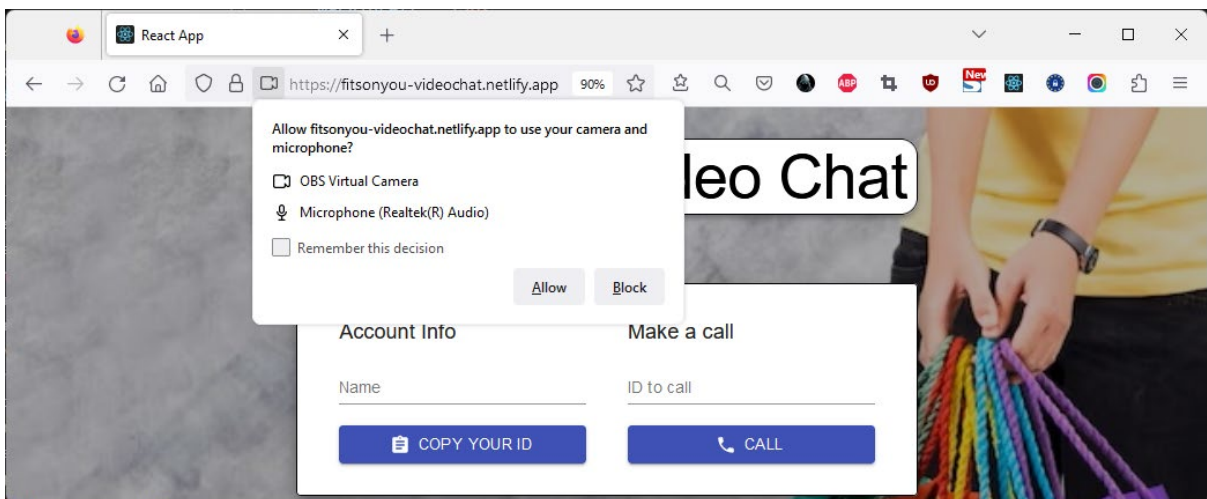


Figure 40. Enabling to use camera and microphone by browser

2.4. Graphical User Interface (GUI)

2.4.1. Client section

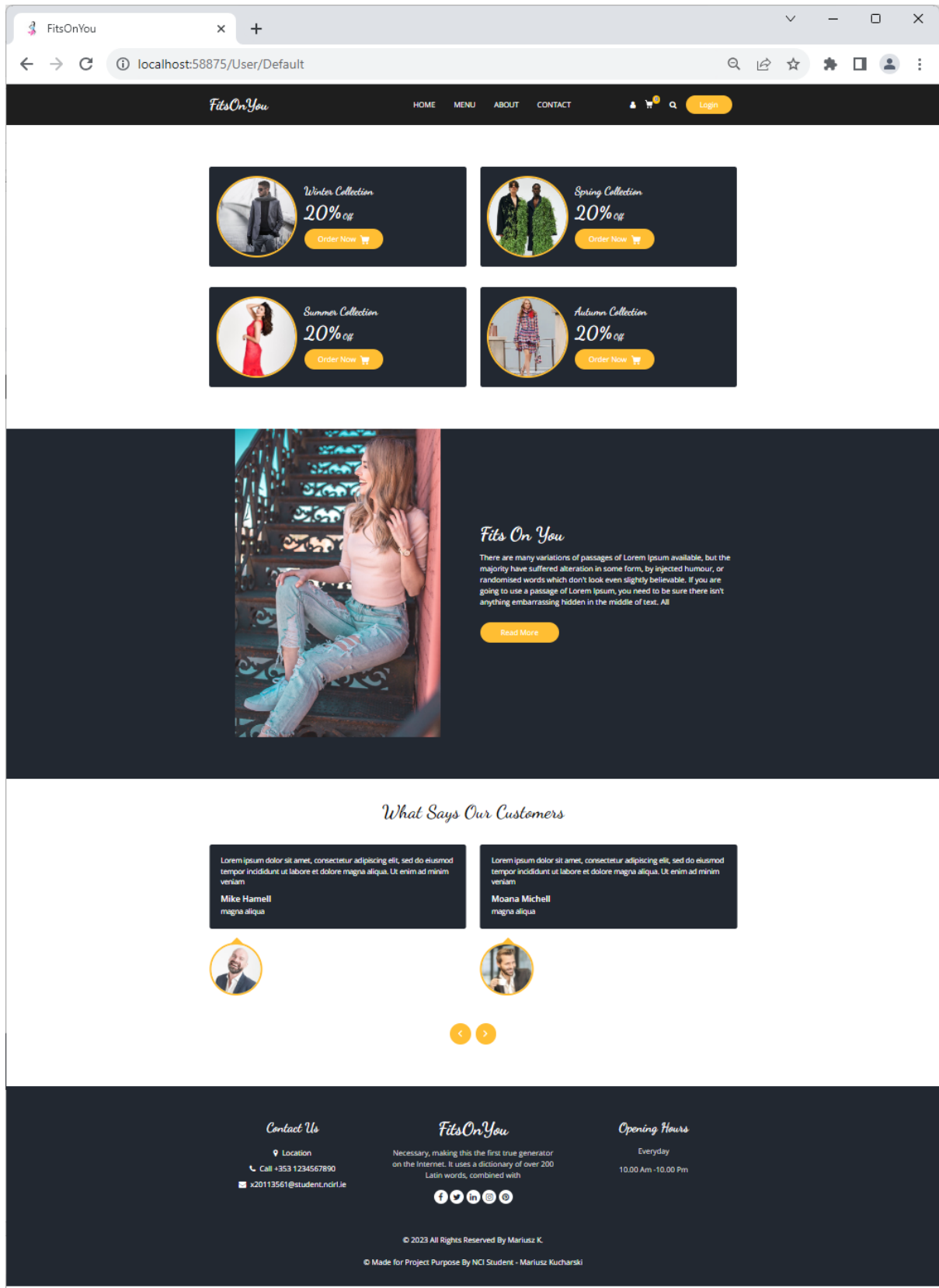


Figure 41. Main page of the application

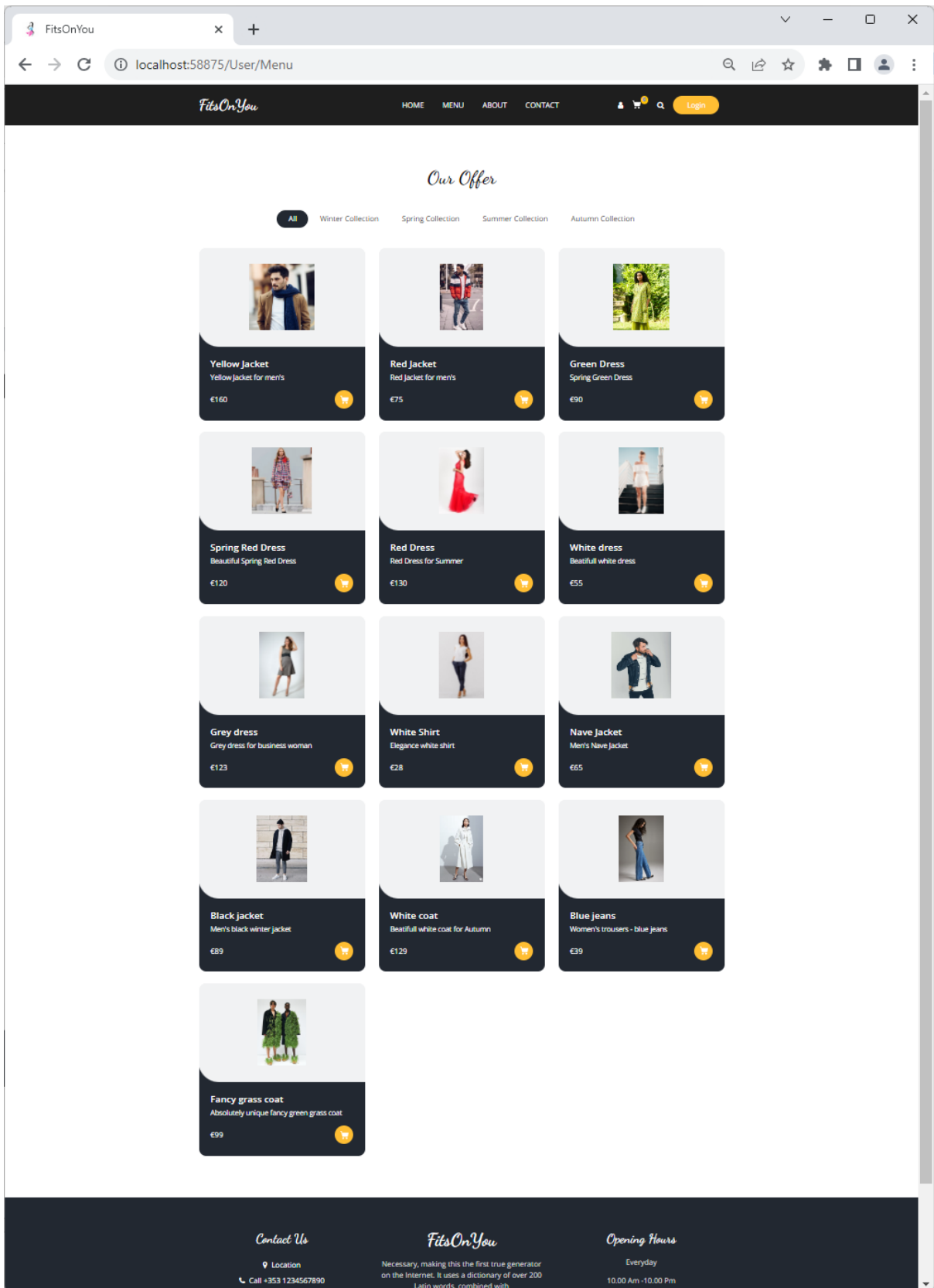


Figure 42. Store's offer menu page

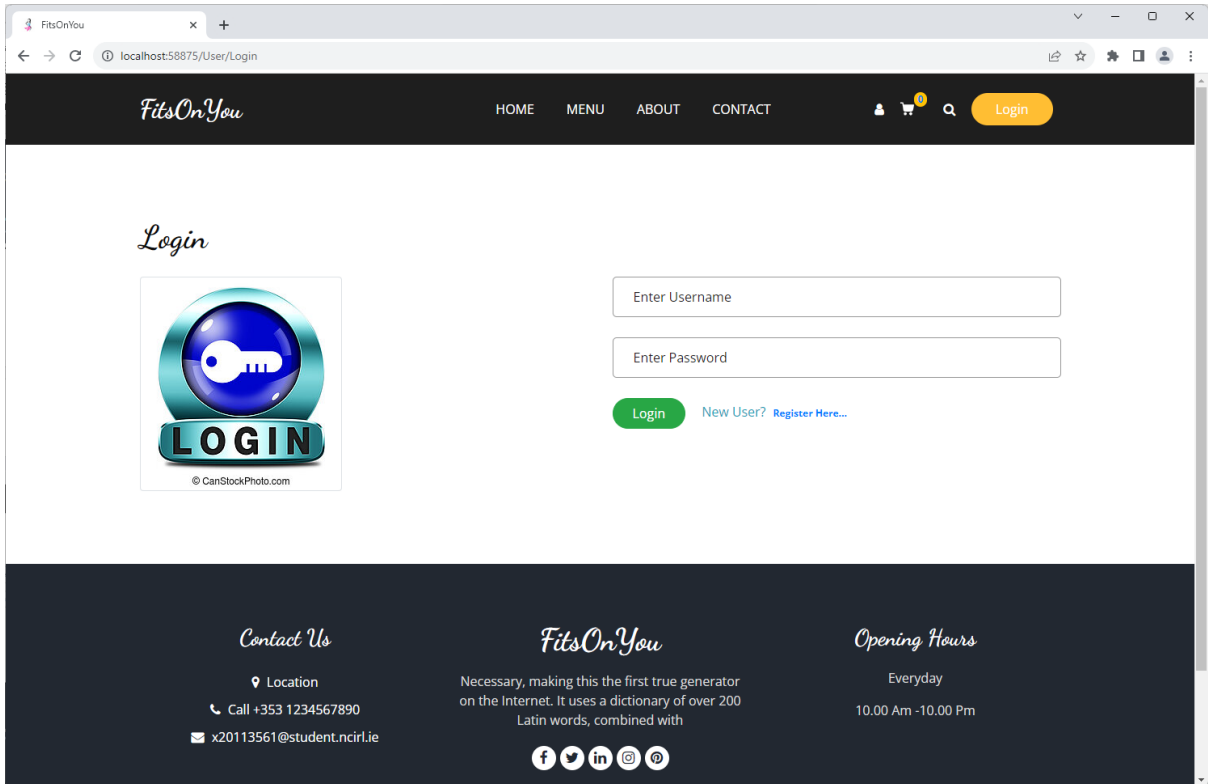


Figure 43. Login page for registered users

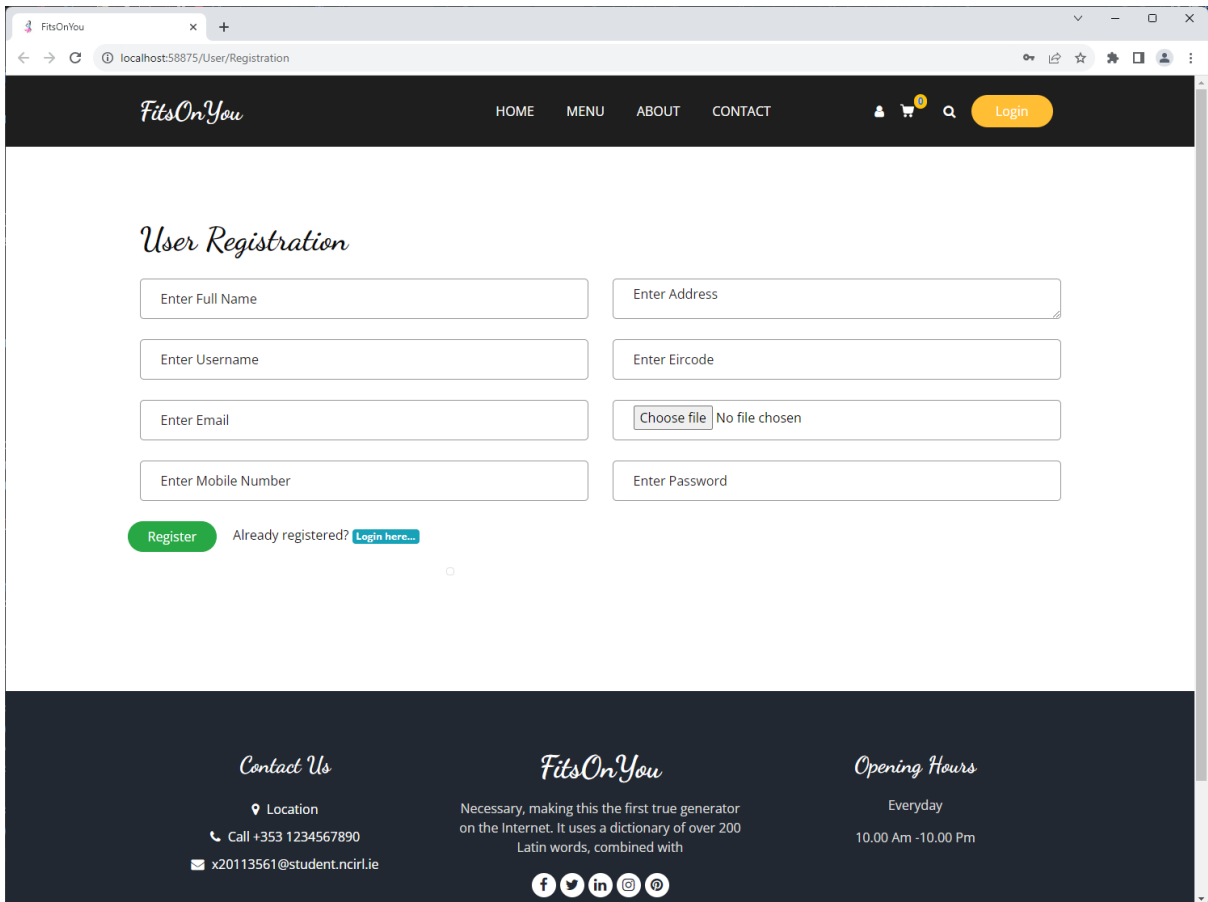


Figure 44. Registration form for a new customer

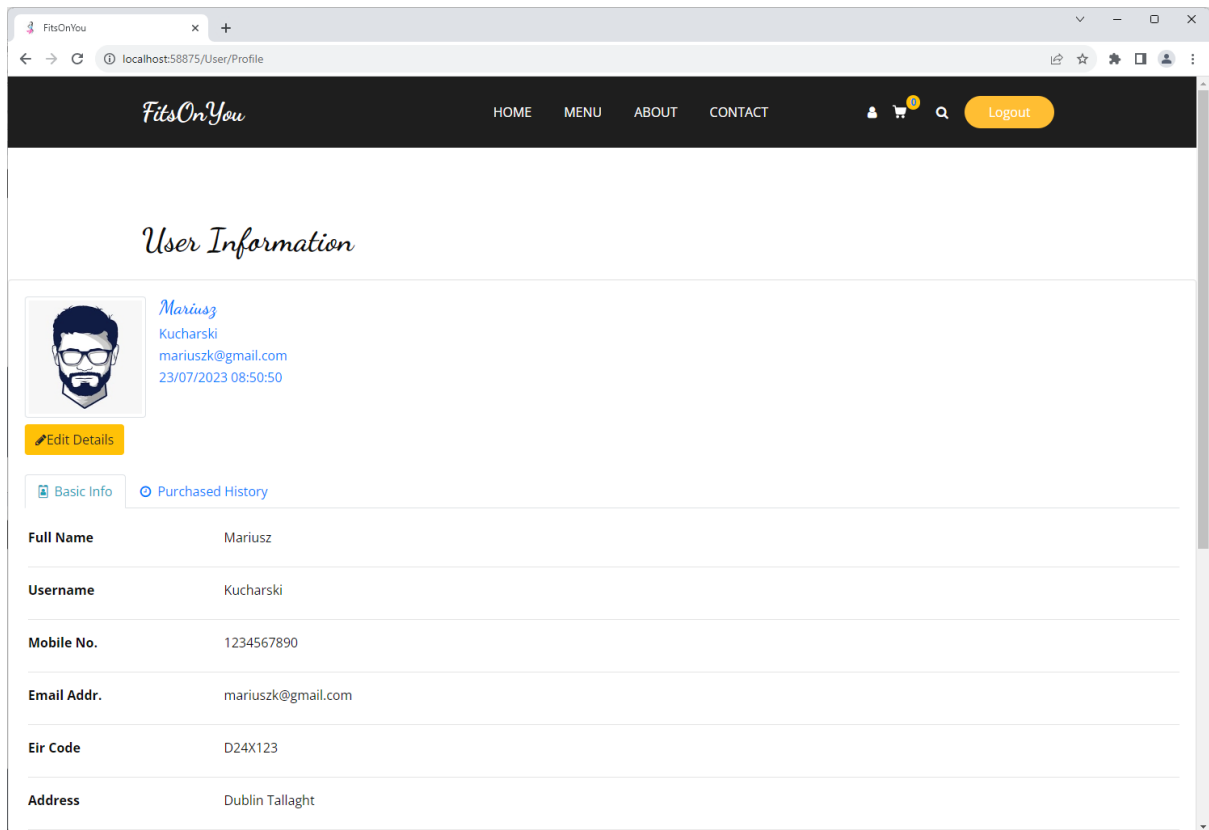


Figure 45. User profile page

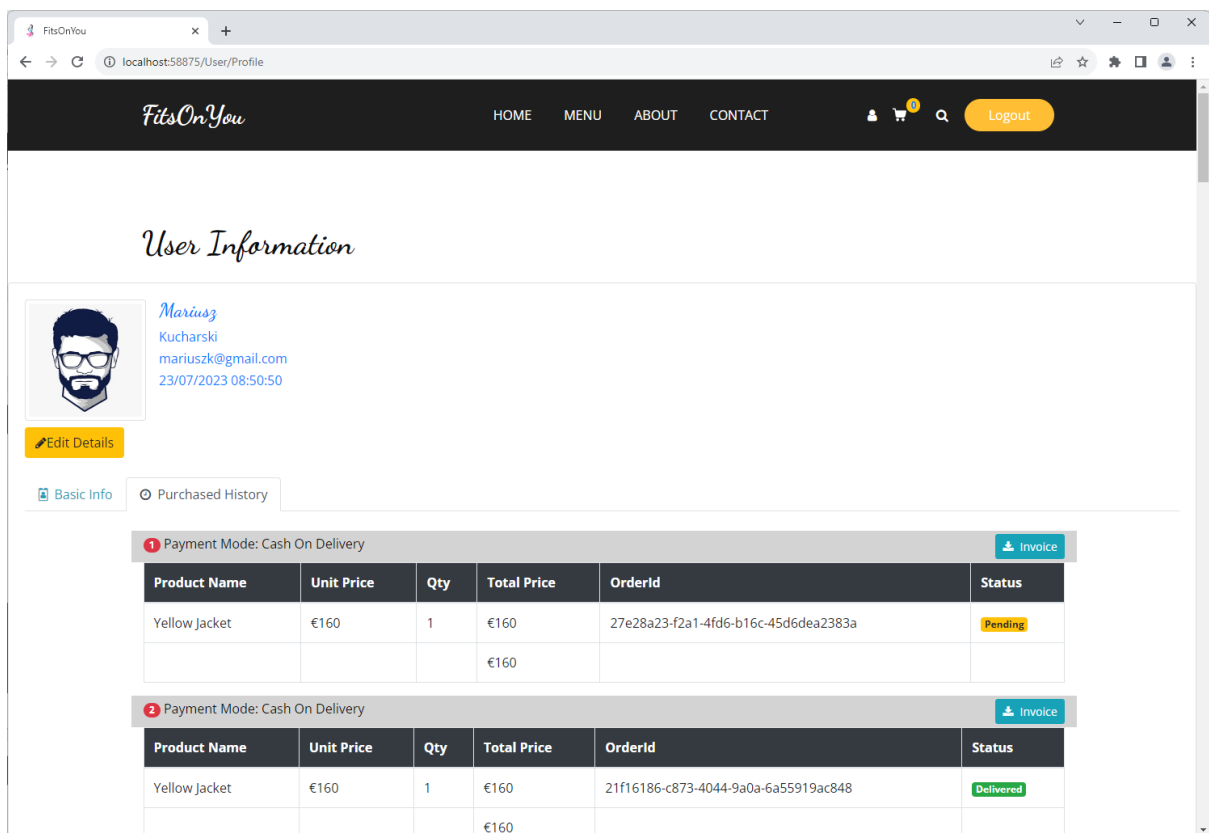


Figure 46. User's purchase history

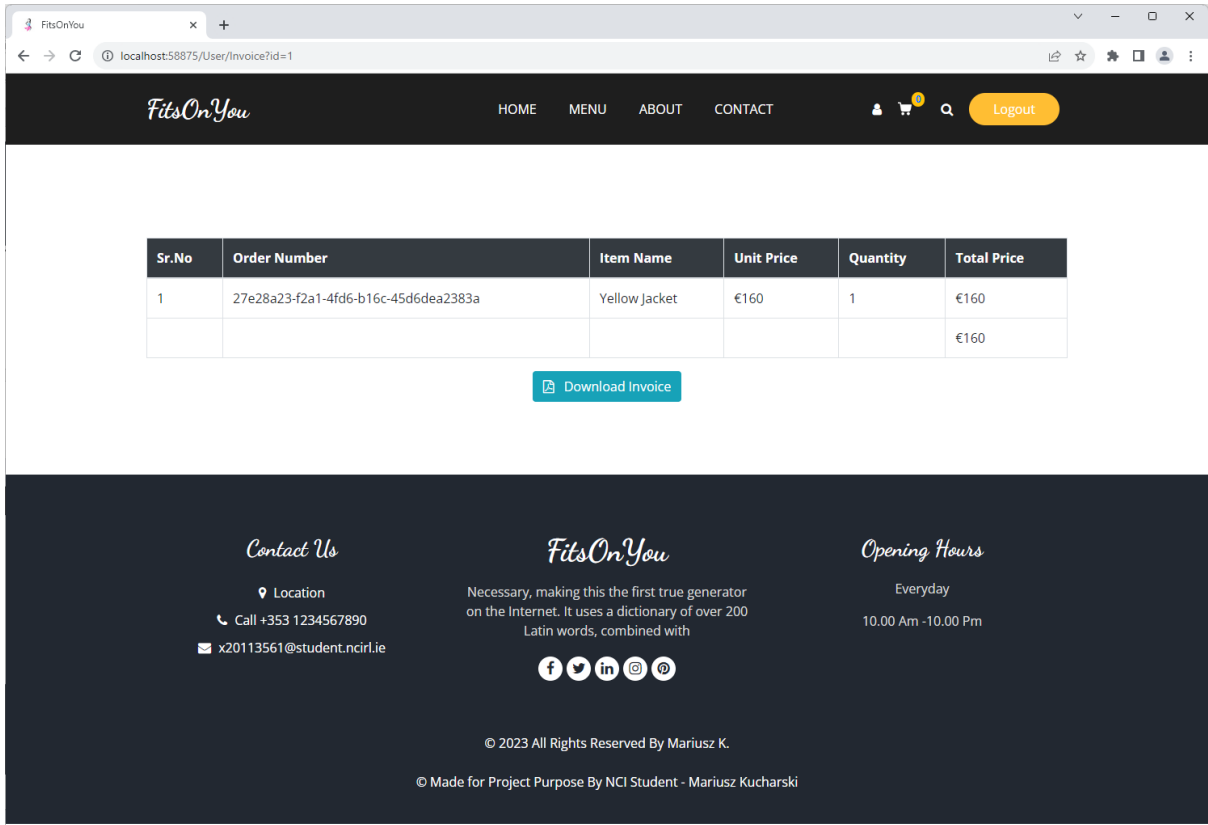


Figure 47. Details of the order after click Invoice from purchased history

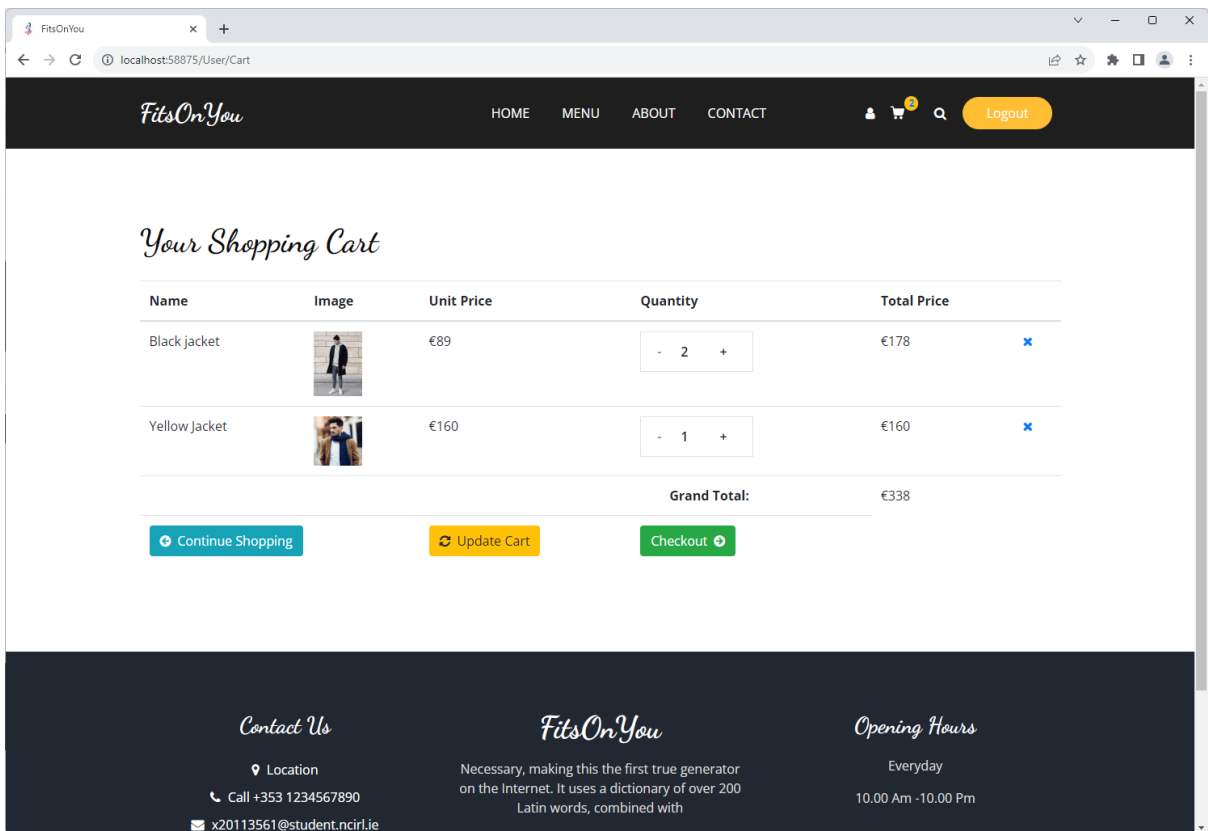


Figure 48. Shopping cart page

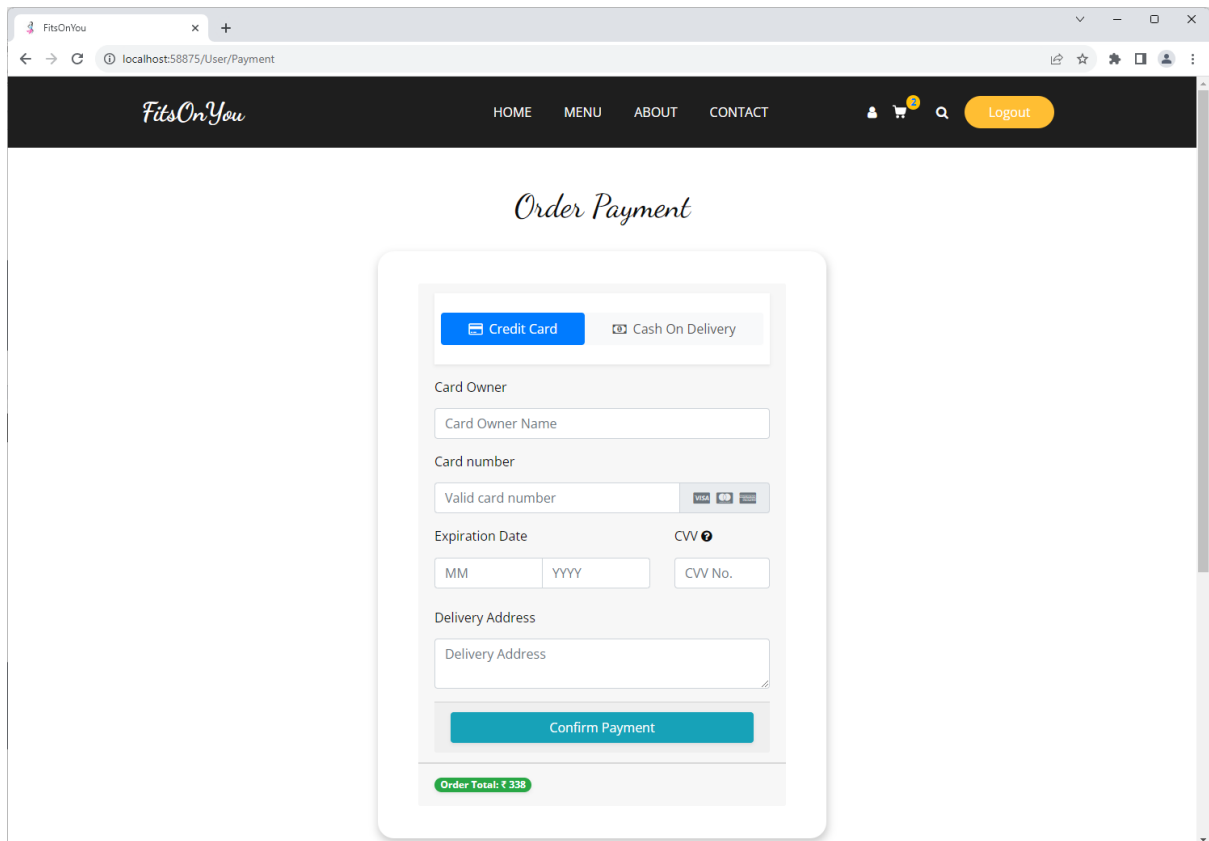


Figure 49. Order payment – Credit Card option page

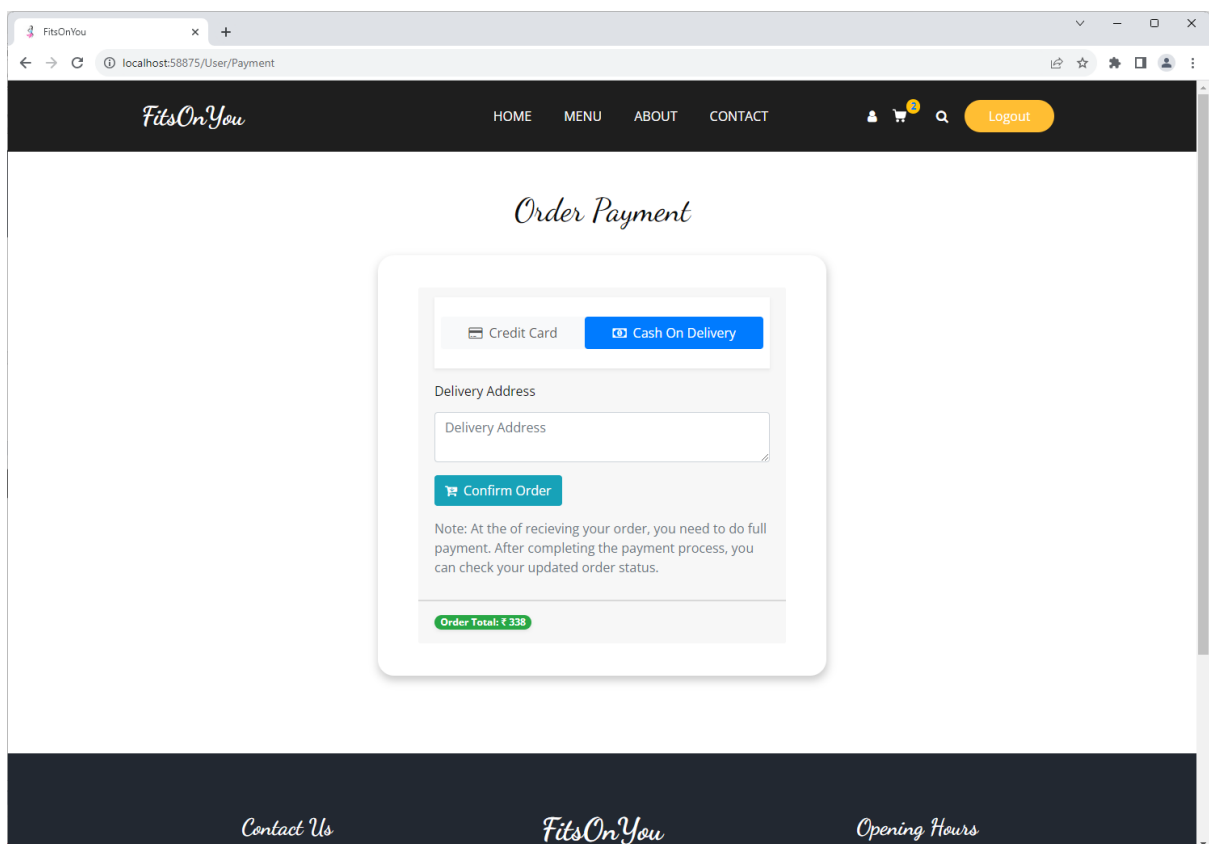


Figure 50. Order payment – Cash on Delivery option page

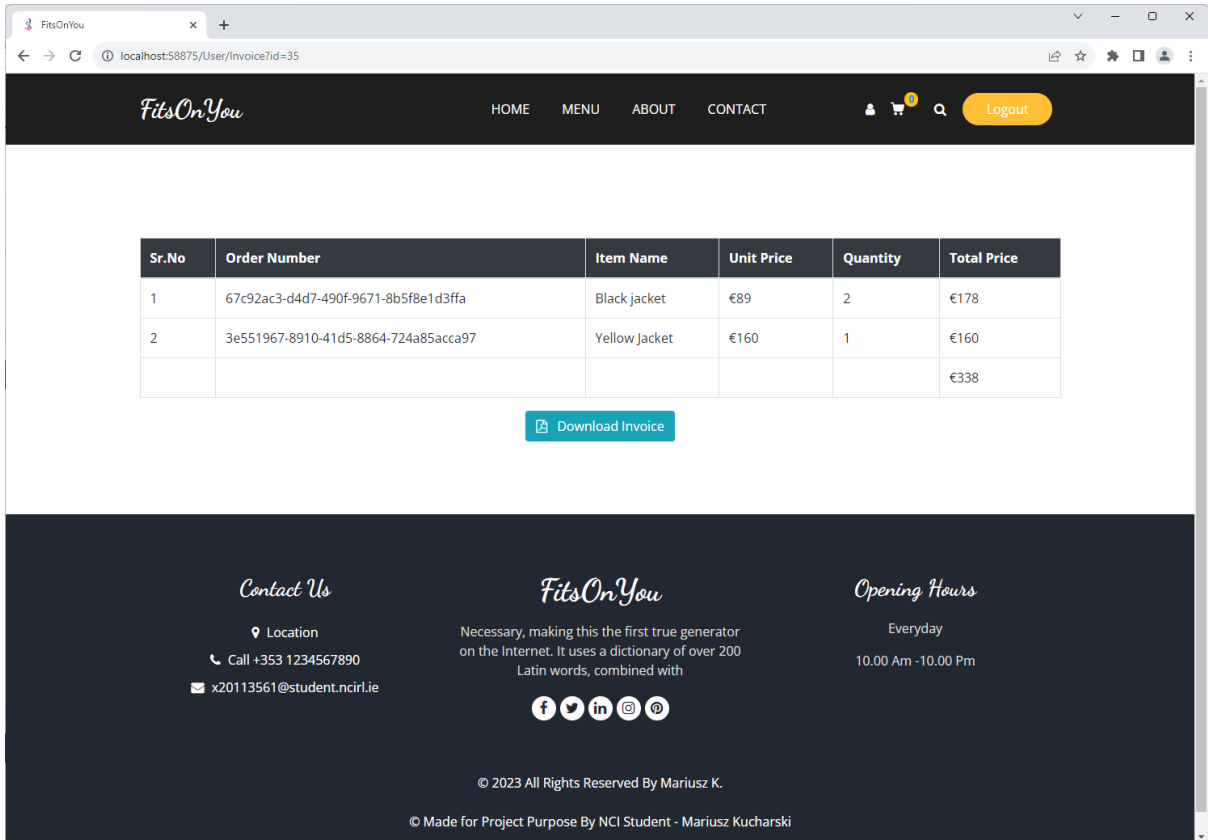


Figure 51. Insight in the invoice after accepting the payment option

2.4.2. Admin section

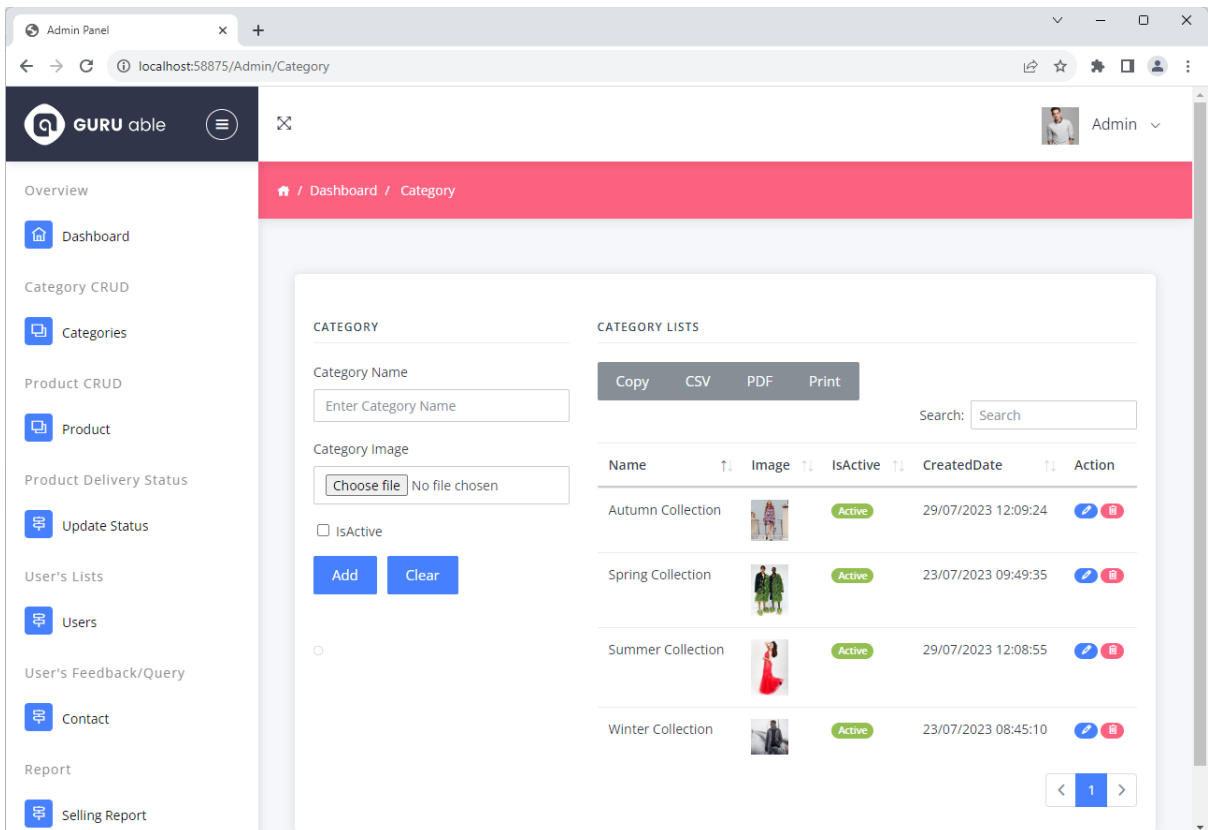


Figure 52. Administration panel – Managing of categories of products page

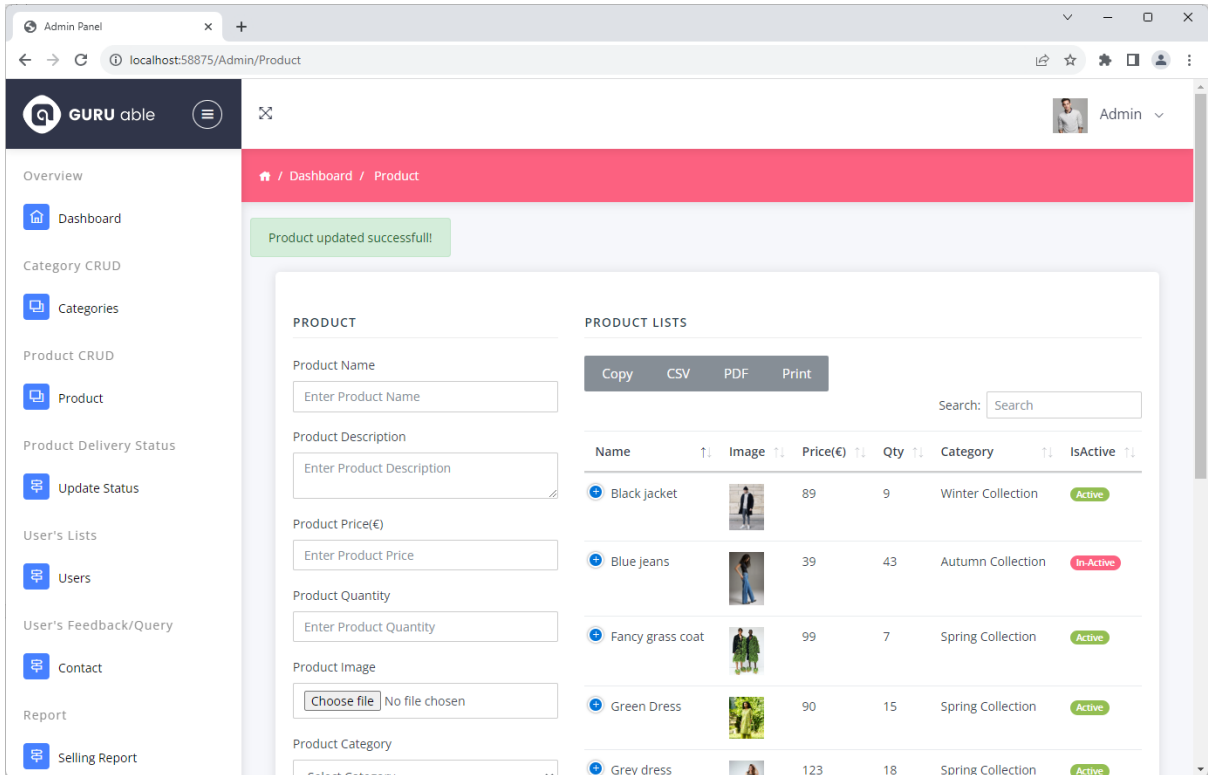


Figure 53. Administration panel – Managing of store’s offer and products page

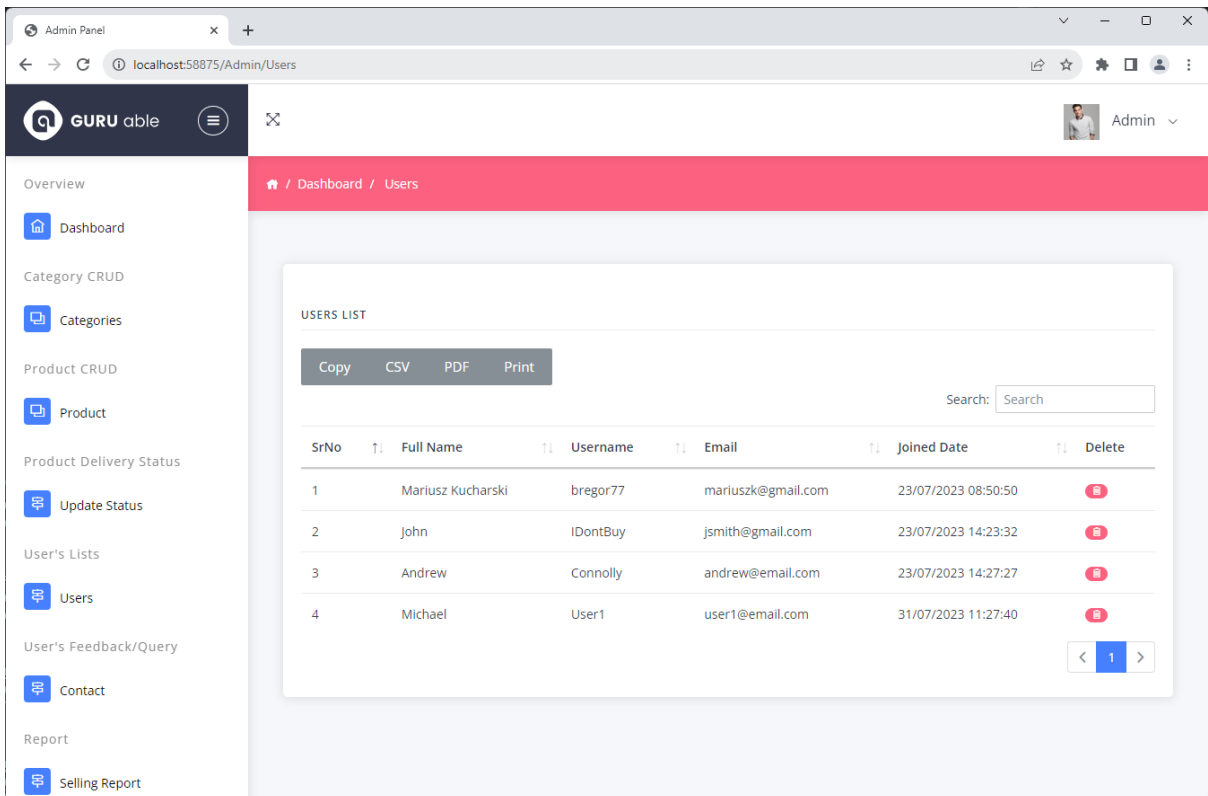


Figure 54. List of the registered client’s page

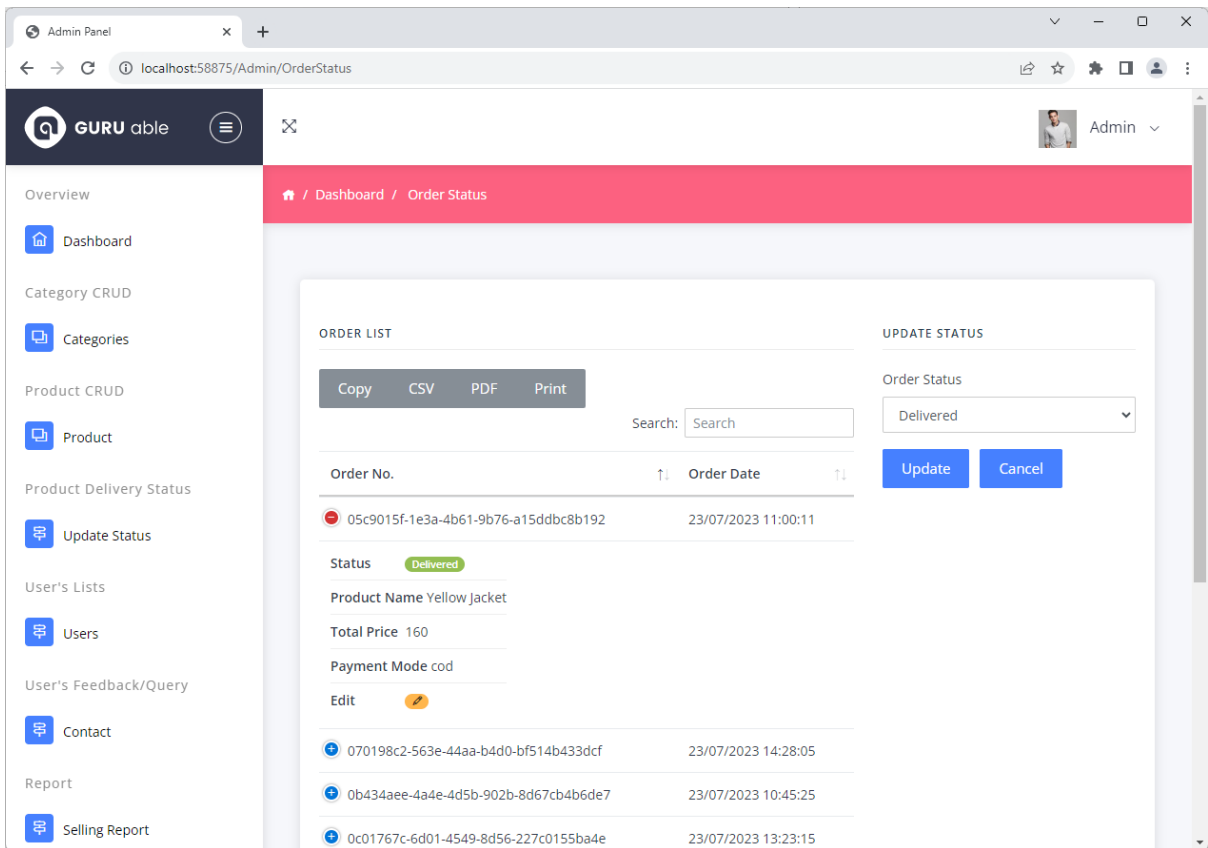


Figure 55. Settings of orders status page

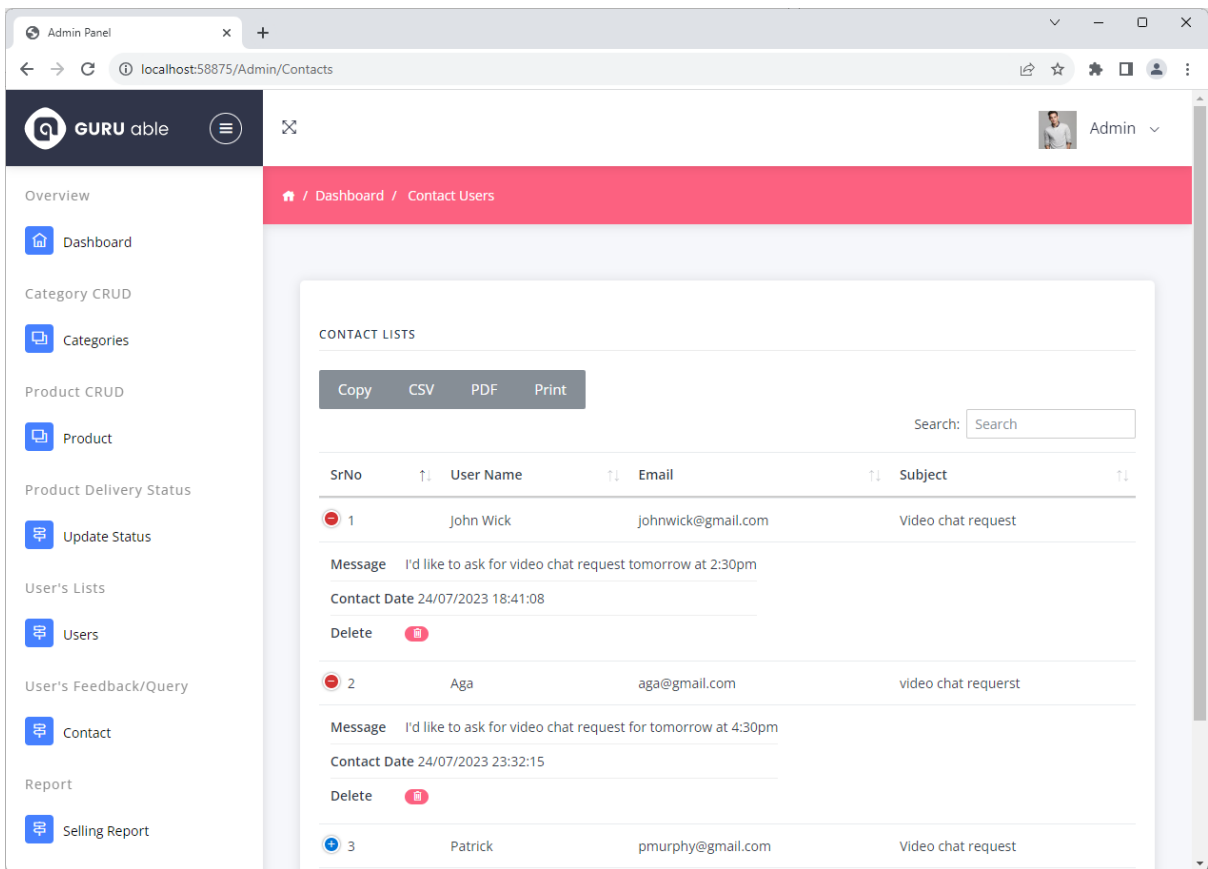


Figure 56. The list of information from user's page

All sections and all information contained in tabs in the Admin section have the option of printing, saving to pdf, saving to a CSV file (possible to open and preview in MS Excel), or the option to copy the version. For example, below is a list of products in a printable version.

The screenshot shows a web browser window with two tabs labeled 'Admin Panel'. The main content area displays a table of products with columns for Name, Image, Price(K), Qty, Category, IsActive, Description, and CreatedDate. A print dialog box is open on the right side, showing options for Destination (Save as PDF), Pages (All), and Layout (Portrait). Below the table, a row for 'Yellow Jacket' is visible with details: 160, 7, Winter Collection, Active, Yellow Jacket for men's, and 23/07/2023 08:49:28.

Name	Image	Price(K)	Qty	Category	IsActive	Description	CreatedDate
Black jacket		89	9	Winter Collection	Active	Men's black winter jacket	23/07/2023 14:49
Blue jeans		39	43	Autumn Collection	Active	Women's trousers - blue jeans	23/07/2023 15:06
Fancy grass coat		99	7	Spring Collection	Active	Modestly unique fancy green grass coat	23/07/2023 15:06
Green Dress		99	15	Spring Collection	Active	Spring Green Dress	23/07/2023 09:30
Grey dress		123	18	Spring Collection	Active	Gray dress for business woman	23/07/2023 14:44
Nave jacket		45	26	Autumn Collection	Active	Men's Nave jacket	23/07/2023 14:44
Red Dress		130	25	Summer Collection	Active	Red Dress for Summer	23/07/2023 12:13
Red jacket		75	12	Winter Collection	Active	Red jacket for women's	23/07/2023 09:59
Spring Red Dress		120	5	Autumn Collection	Active	Beautiful Spring Red Dress	23/07/2023 09:59
White coat		129	19	Autumn Collection	Active	Beautiful white coat for Autumn	23/07/2023 14:58
White dress		55	32	Summer Collection	Active	Beautiful white dress	23/07/2023 14:42
White Shirt		28	45	Spring Collection	Active	Elegant white shirt	23/07/2023 14:45
Yellow jacket		160	7	Winter Collection	Active	Yellow jacket for men's	23/07/2023 08:49

Figure 57. The printable version of product's list

The screenshot shows a web browser window with a single tab labeled 'FitsOnYou'. The address bar shows 'localhost:58875/User/About'. The page features a dark background with the 'Fits On You' logo in a cursive font. A navigation menu includes 'HOME', 'MENU', 'ABOUT', and 'CONTACT'. A 'Login' button is visible in the top right corner. The main content area features a large image of a woman sitting on a staircase, wearing a light pink top and light blue jeans. To the right of the image, the text reads 'Fits On You' followed by a paragraph of Lorem Ipsum text and a 'Read More' button.

Figure 58. About page

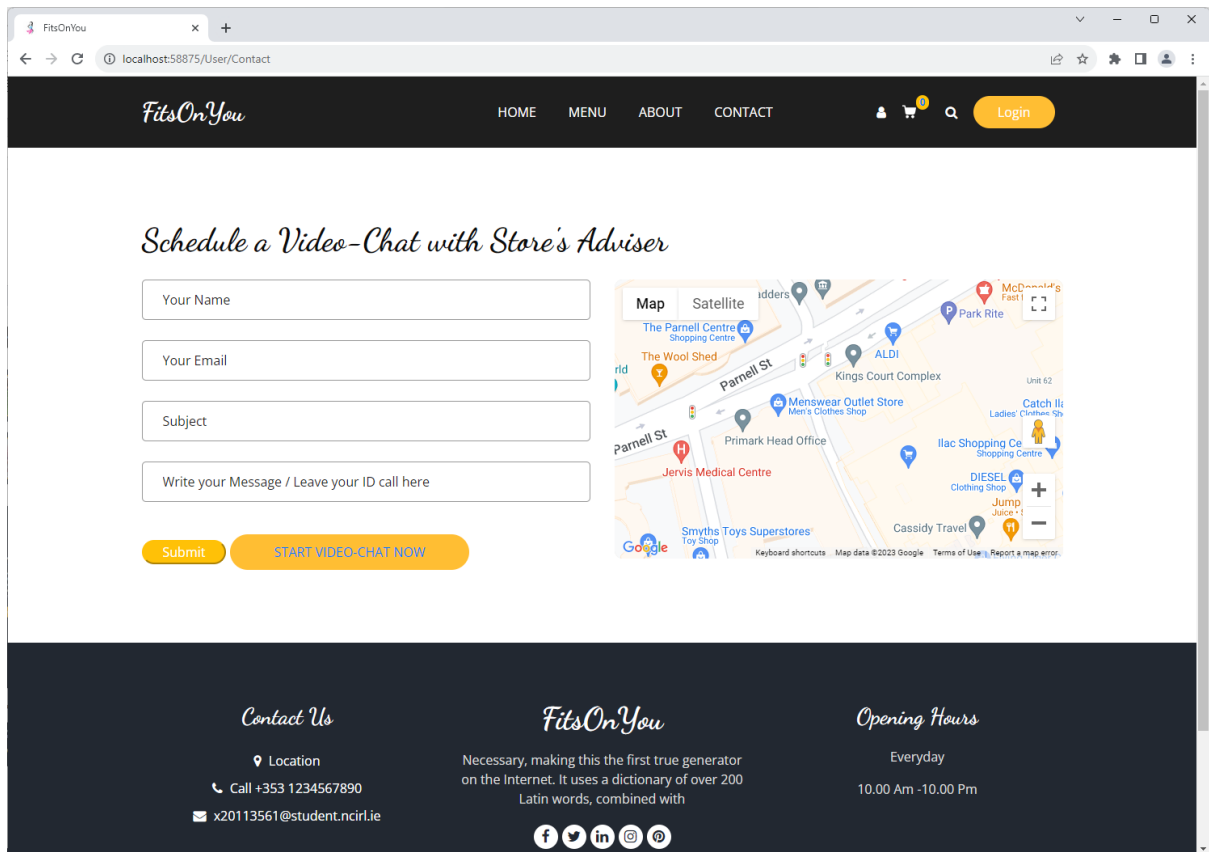


Figure 59. Contact page with video-chat request option and link to the video-chat app

A short description of how video-chat works

In the video-chat application, just click on the button to generate a unique code, used to identify a secure video connection. The patient receives the code generated in this way by e-mail, and then logs in and paste the previously generated code into "ID to call" field. Depending on the needs, the connection can be initiated in both directions between patient and doctor. As you can see here, connection is possible using various devices; in this case it's a PC and mobile phone.

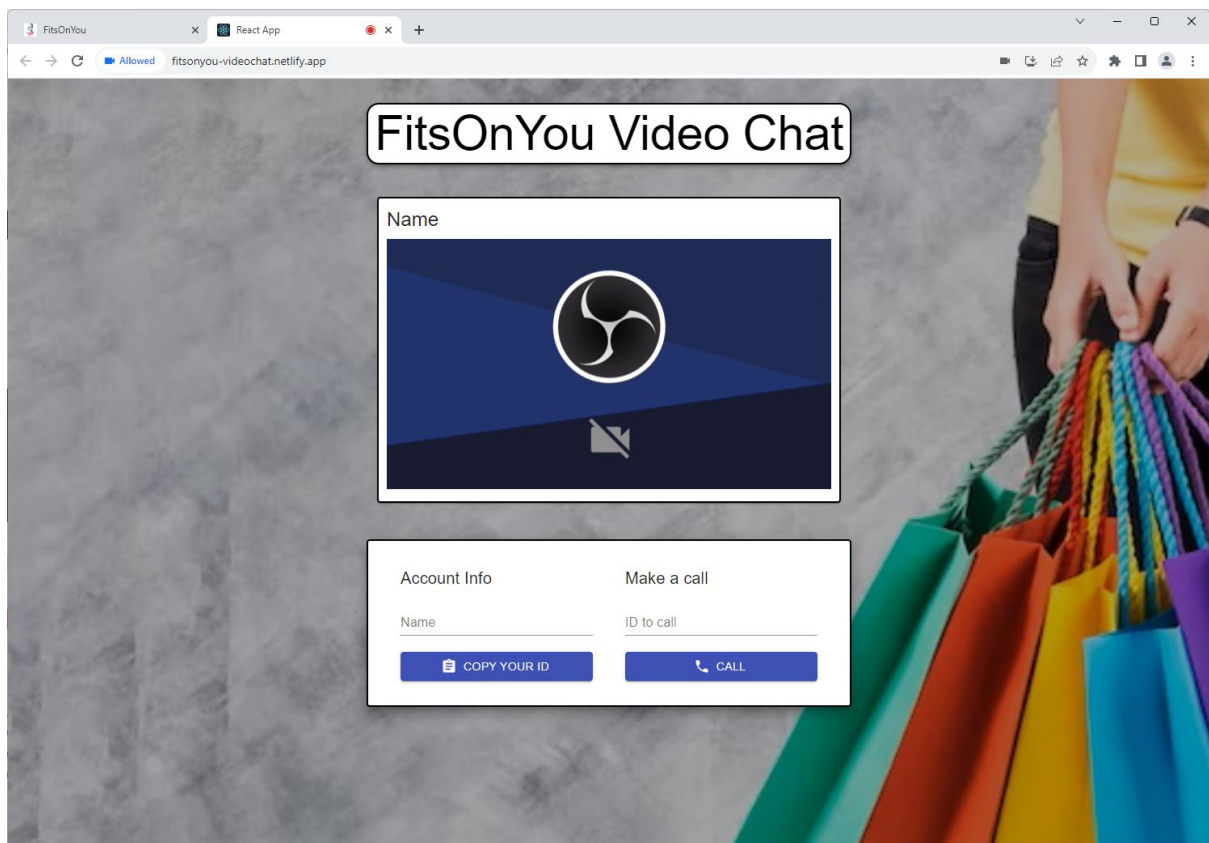


Figure 60. View of the Video-chat application

Stock market analytics

The stock market analytics section – prototype version

Access to the store's sales reports and stock market analyzes and diagrams will be carried out under the Selling Reports and Analytics button in the Administrator section of application.

This section is in the prototype version and has not yet been implemented in the main code of the application. However, to generate the diagrams shown below, I used Python code with simple use of the **yfinance** libraries. The code is in a working version and can be run in VS Code or any environment that supports Python. I've also added the code files to my repository on GitHub.

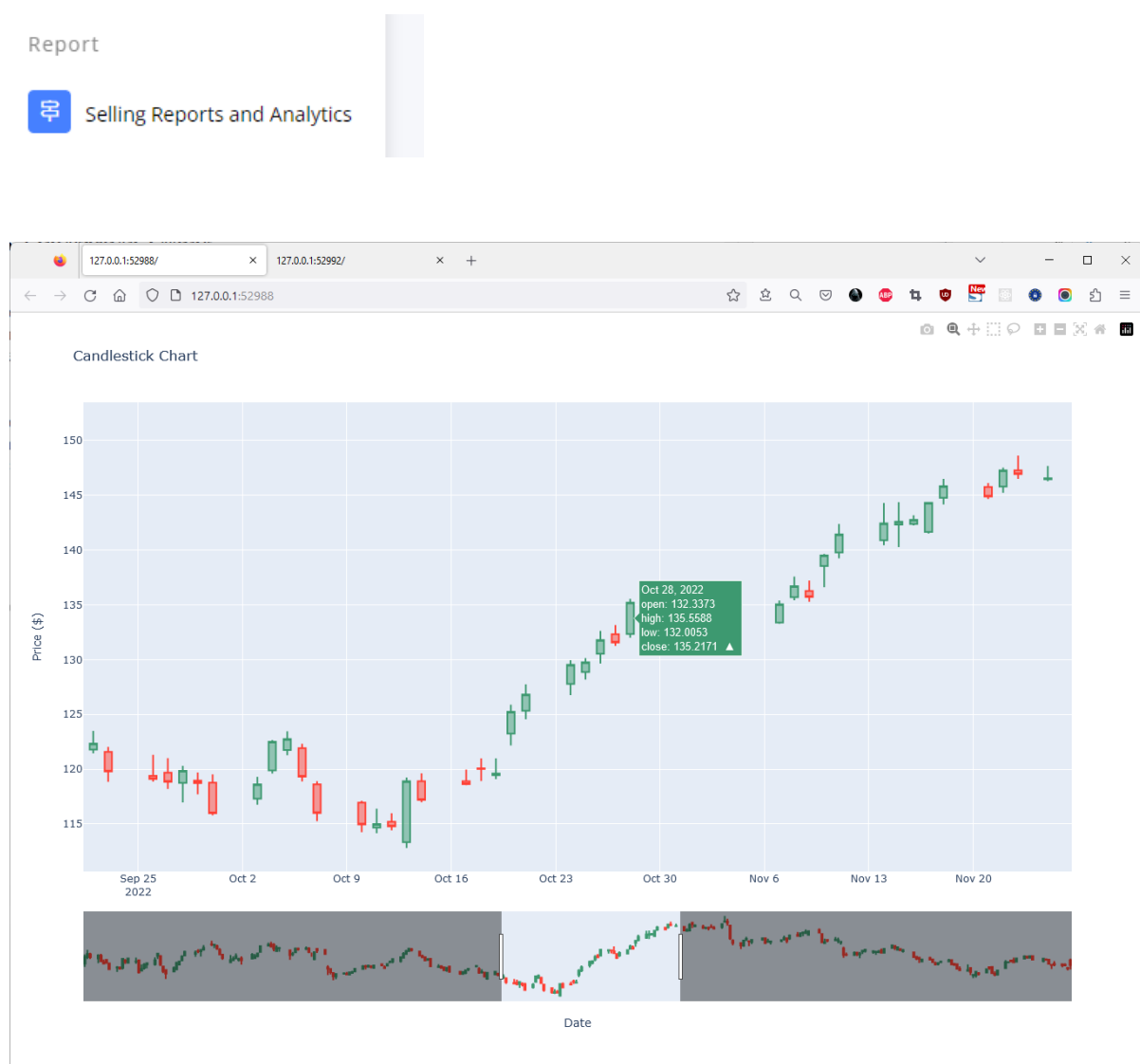


Figure 61. Display a stock market data

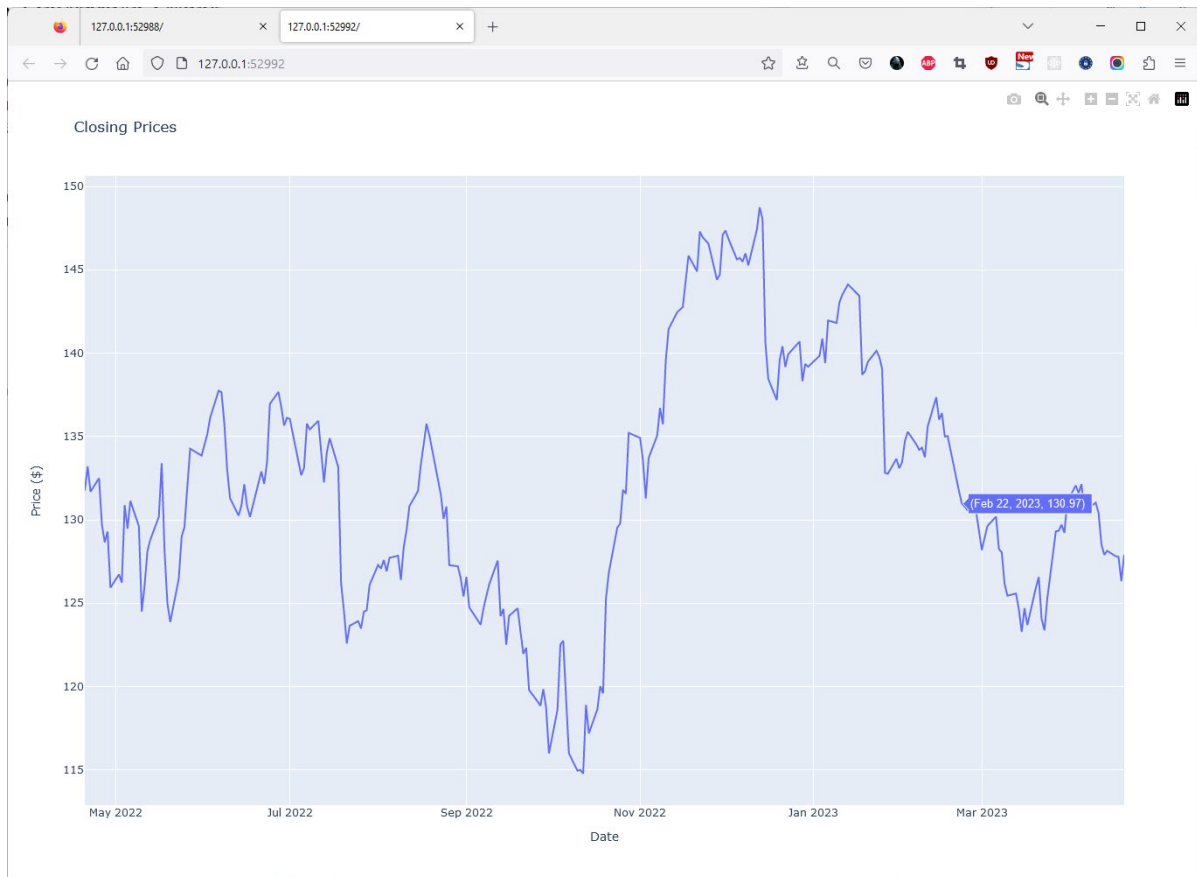


Figure 62. Display a stock market information of selected company

2.5. Testing

The tests can be of different types, such as unit tests, integration tests, and end-to-end tests, each serving a specific purpose in ensuring the application functions correctly and meets the requirements.

- **Unit Tests:** These tests focus on individual units of code, typically testing individual methods or small parts of a page. They are designed to validate the correctness of the code logic and isolate potential issues to specific code blocks.
- **Integration Tests:** These tests verify the interaction between different components, modules, or layers within the application. They ensure that these parts work together as expected and that they integrate seamlessly.
- **End-to-End Tests:** These tests simulate real user interactions with the web application by interacting with the web pages through a web browser or automated browser testing tools. They are more comprehensive and check if the entire application flow works correctly, including user input, data processing, and response output.

How ASP.NET WebForms tests work:

Testing Framework: ASP.NET WebForms tests are generally written using testing frameworks like MSTest, NUnit or xUnit which provide the necessary infrastructure for creating and executing tests.

Arrange-Act-Assert Pattern: Most test scenarios follow the Arrange-Act-Assert pattern. In the Arrange phase, the test sets up the necessary preconditions, data, and dependencies for the test case. In the Act phase, the test performs the action or triggers the event being tested. Finally, in the Assert phase, the test checks the actual results against expected outcomes.

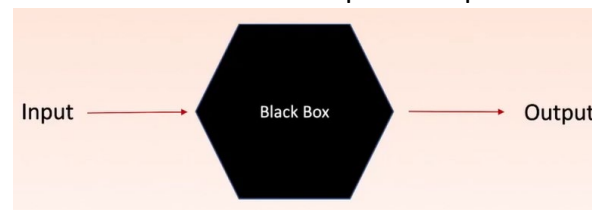
Mocking and Isolation: In some cases, it may be necessary to isolate certain components or dependencies from the test environment. This is achieved using mocking frameworks that create fake implementations of certain parts of the application to simulate specific behaviours.

Web UI Testing: For end-to-end tests, tools like Selenium WebDriver are commonly used. Selenium allows testers to automate the browser, interact with web elements, and perform actions like clicking buttons, entering text, and verifying page content.

Continuous Integration: ASP.NET WebForms tests can be integrated into the continuous integration/continuous deployment (CI/CD) pipeline to automatically run tests whenever changes are made to the codebase, ensuring that new updates do not introduce regressions. For CI/CD, using the CircleCI platform is a very good choice. Here we can easily configure a config.yml file containing a workflow with instructions in YAML for build, testing and deploying our project, e.g., to Azure server.

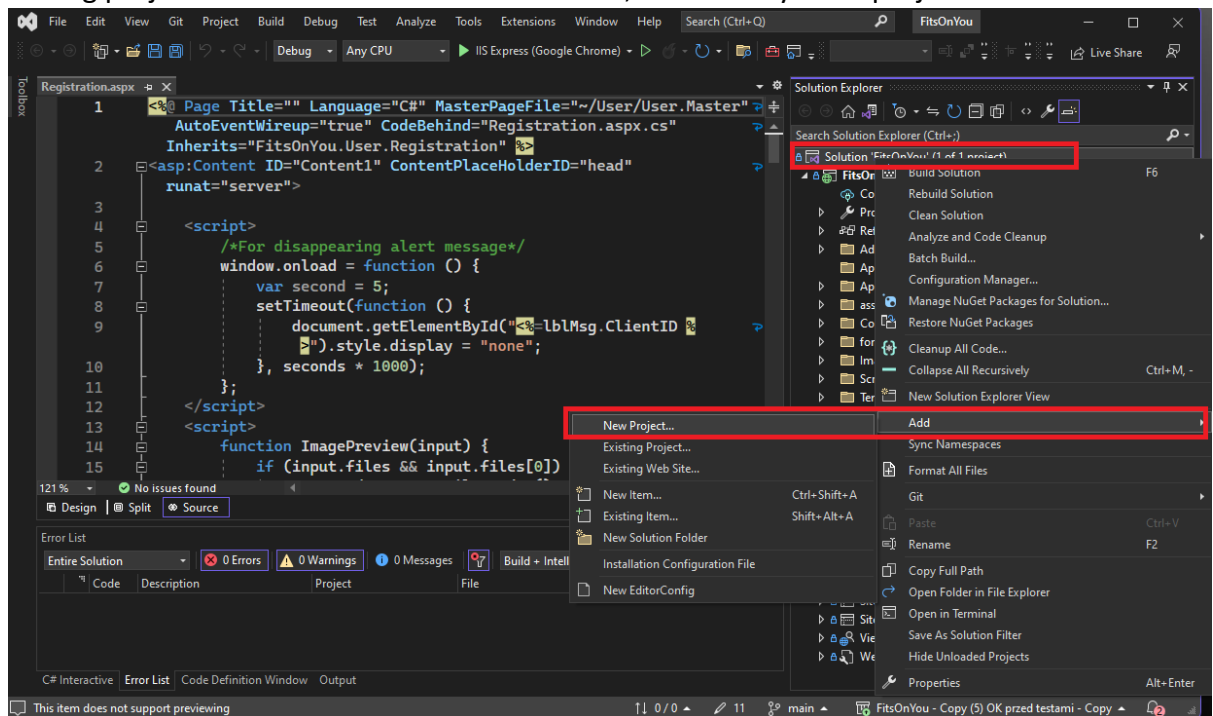
2.5.1. How I performed Unit Tests

For testing purpose of unit tests for demonstrate a unit tests for ASP.NET WebForms, I created a new project withing a Solution of my main project. We can say that unit test work when we have an input – output scenario. Usually it's called a black box tests.

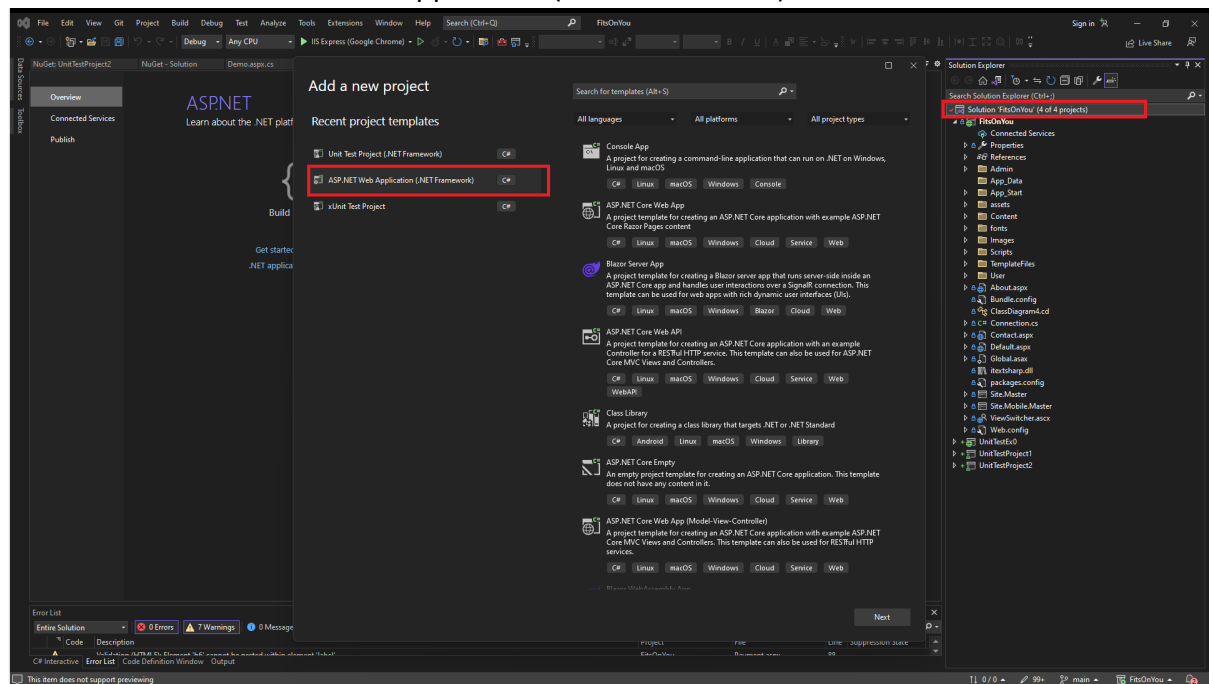


It means, we don't have in our logic an independences like database queue, external API's, etc. These type of tests are perfect when we test algorithm or if our logic is completely isolated. They should be quick and simple when checking if our algorithms are working correctly. This is what I did, a test of a simple algorithm to check if the entered variables are what they should be in an example of code of ASP.NET Forms.

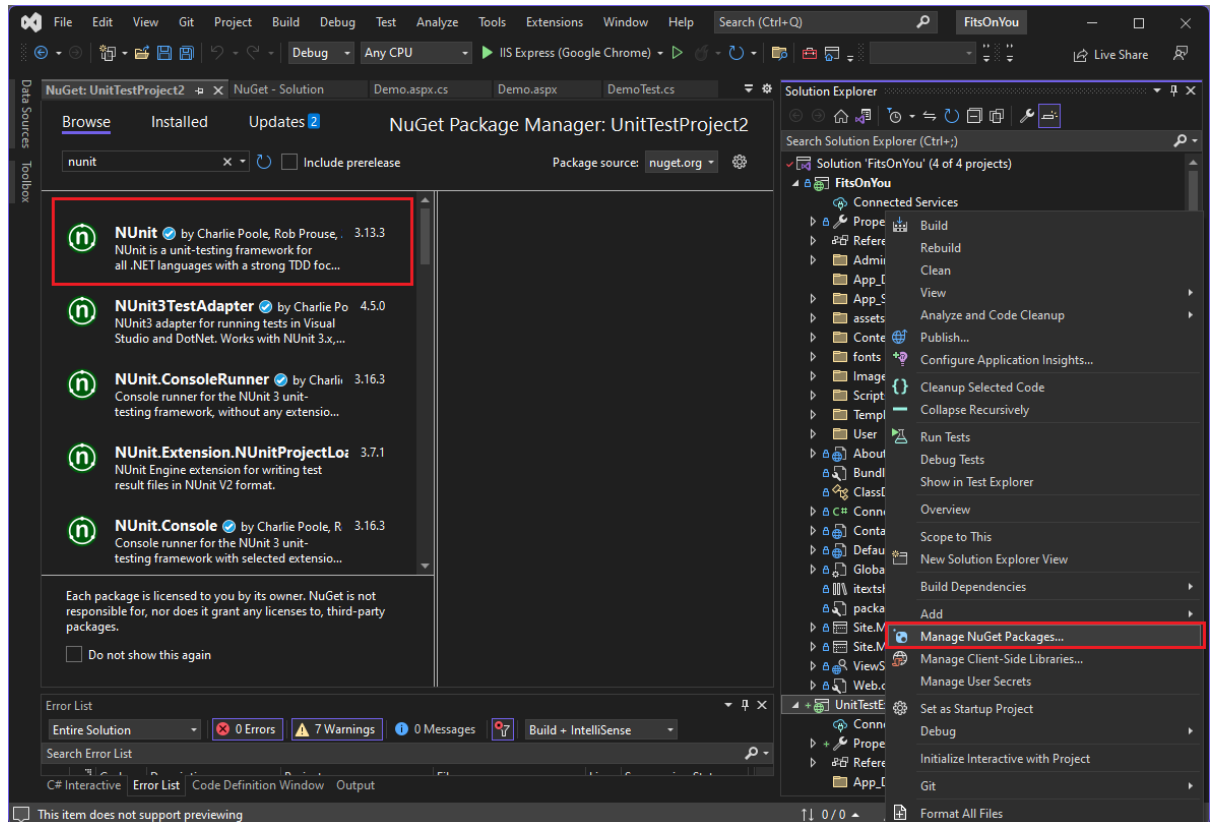
For this purpose, I created a new project inside a solution of my main project. The new testing project also basis on the ASP.NET Forms, same as my main project.



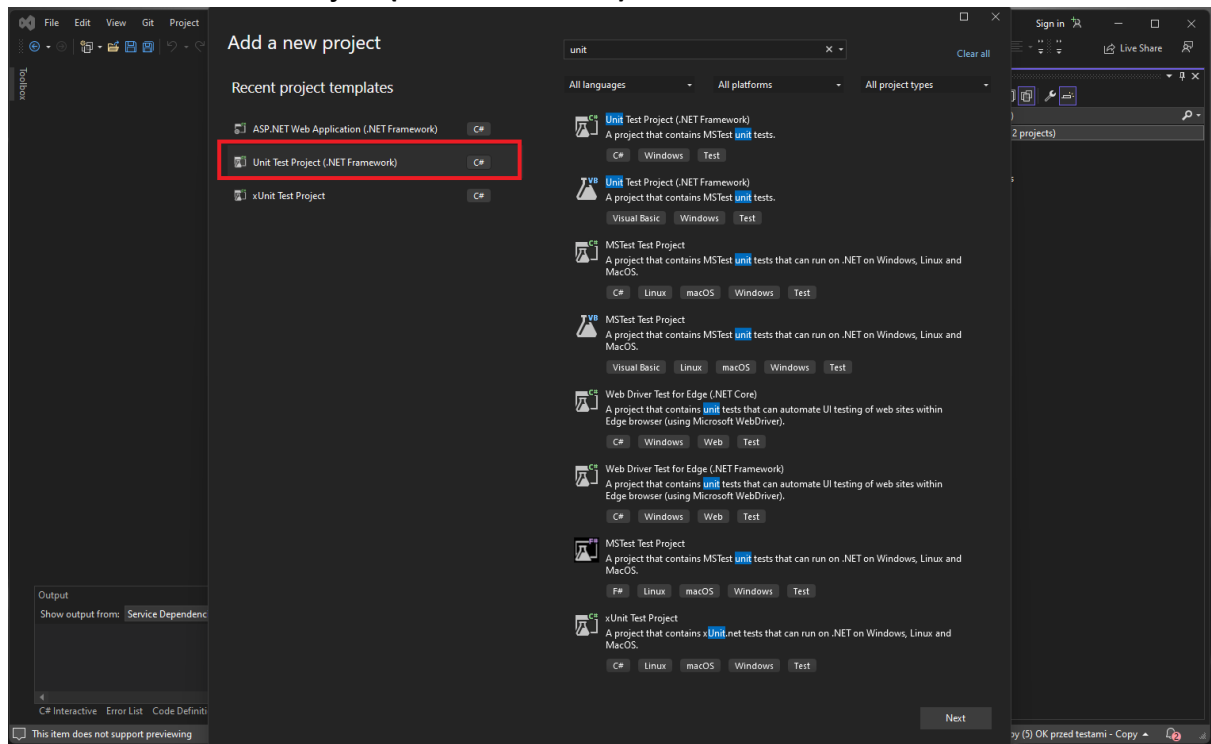
Then I selected ASP.NET Web Application (.NET Framework)



In **NuGet Packages Manager** I installed a **NUnit** framework, one of most popular used for these types of tests.



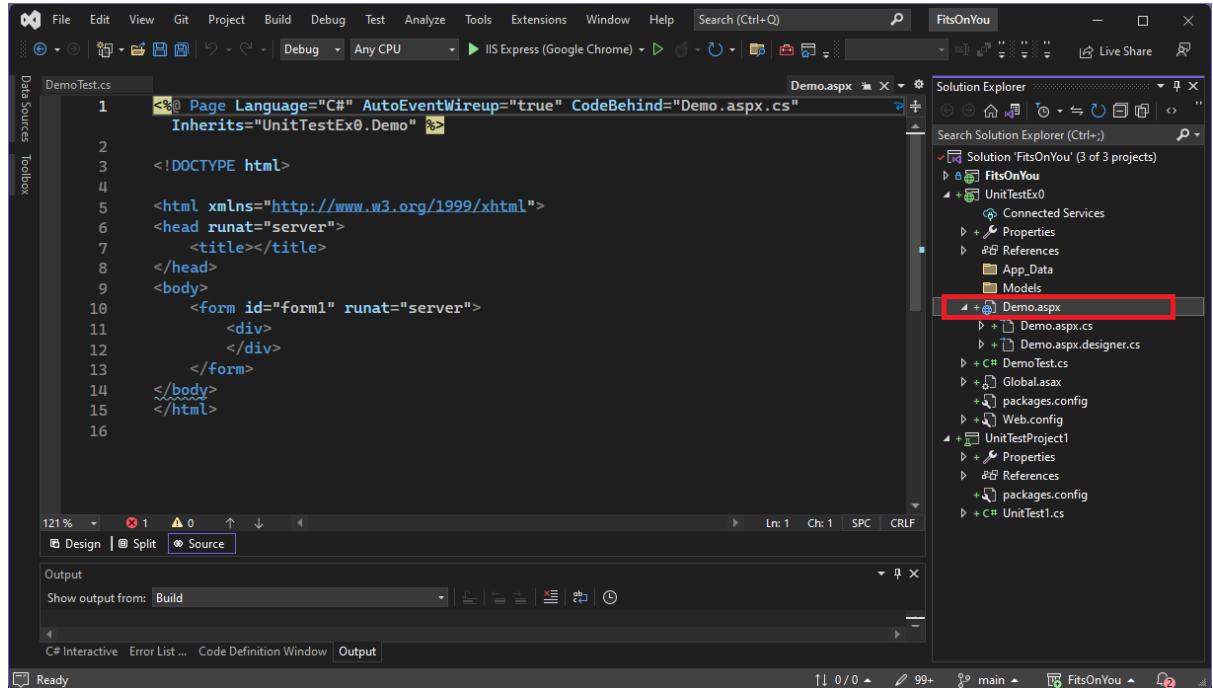
Then, also within the same project solution, I created another project based on the framework **Unit Test Project (.NET Framework)**



So, test project **UnitTestEx0** and **UnitTestProject1** have been created. I also created a simple WEB form **Demo.aspx**, similar to my project except it's more elaborate. However, here we are talking about unit testing, so here we deal with single, simple cases that, for example, are found to be as part of the code as individual cases.

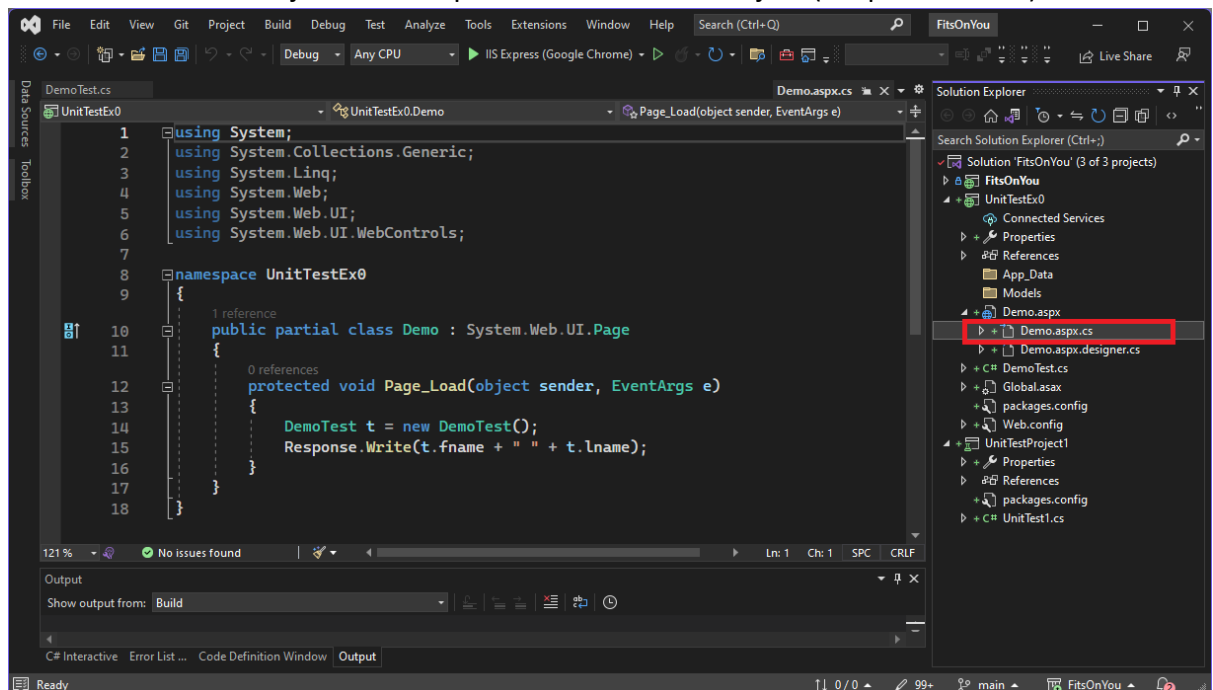
As I discussed ASP.NET earlier, the project contains the front-end and the backend of this frontend.

Frontend looks as below:



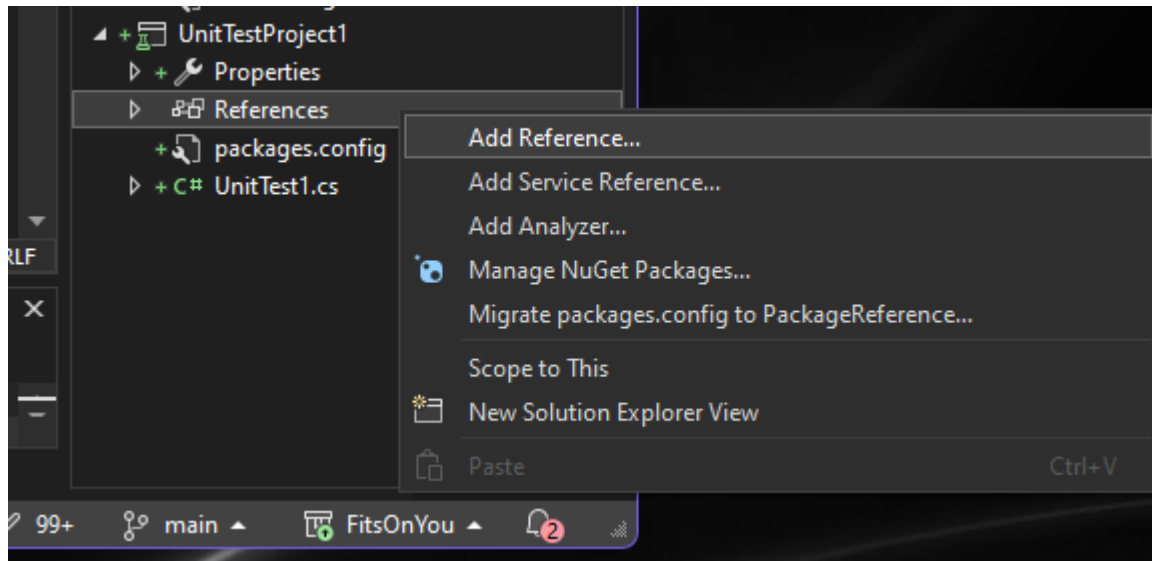
```
1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Demo.aspx.cs" Inherits="UnitTestEx0.Demo" %>
2
3 <!DOCTYPE html>
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7 <title></title>
8 </head>
9 <body>
10 <form id="form1" runat="server">
11 <div>
12 </div>
13 </form>
14 </body>
15 </html>
16
```

In the backend section down below, you can find a **DemoTest** object **t = new DemoTest**. Now we just need to print of value of this object (Response.Write).

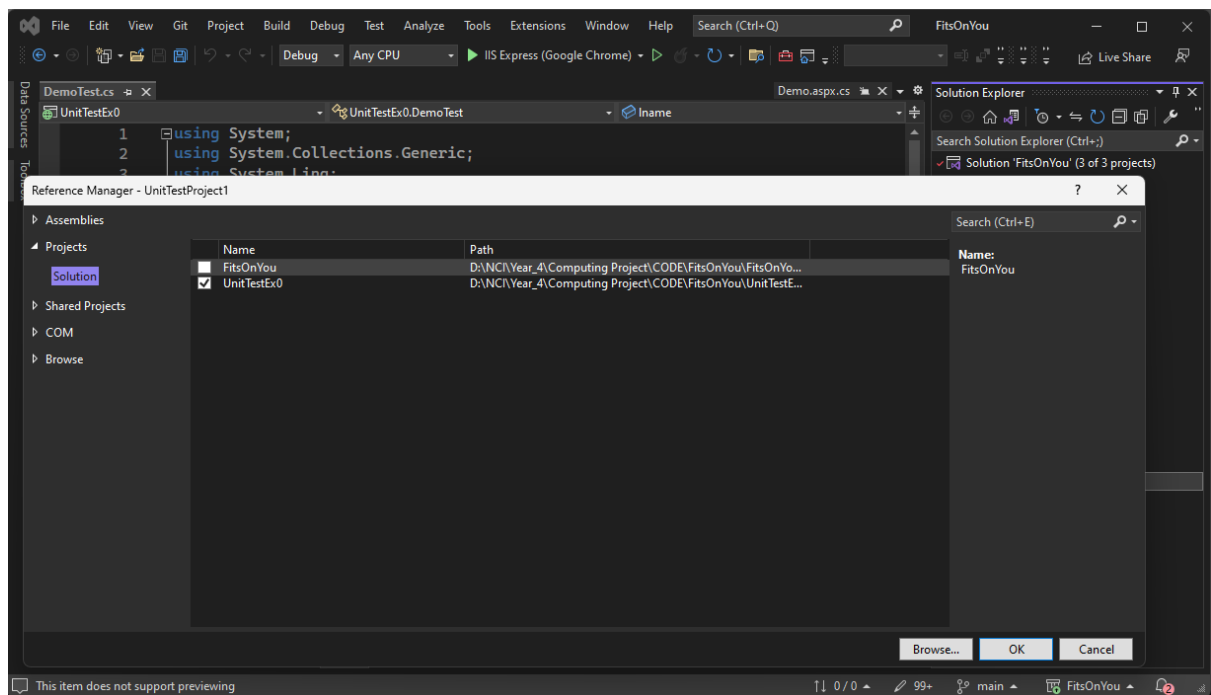


```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.UI;
6 using System.Web.UI.WebControls;
7
8 namespace UnitTestEx0
9 {
10     public partial class Demo : System.Web.UI.Page
11     {
12         protected void Page_Load(object sender, EventArgs e)
13         {
14             DemoTest t = new DemoTest();
15             Response.Write(t.fname + " " + t.lname);
16         }
17     }
18 }
```

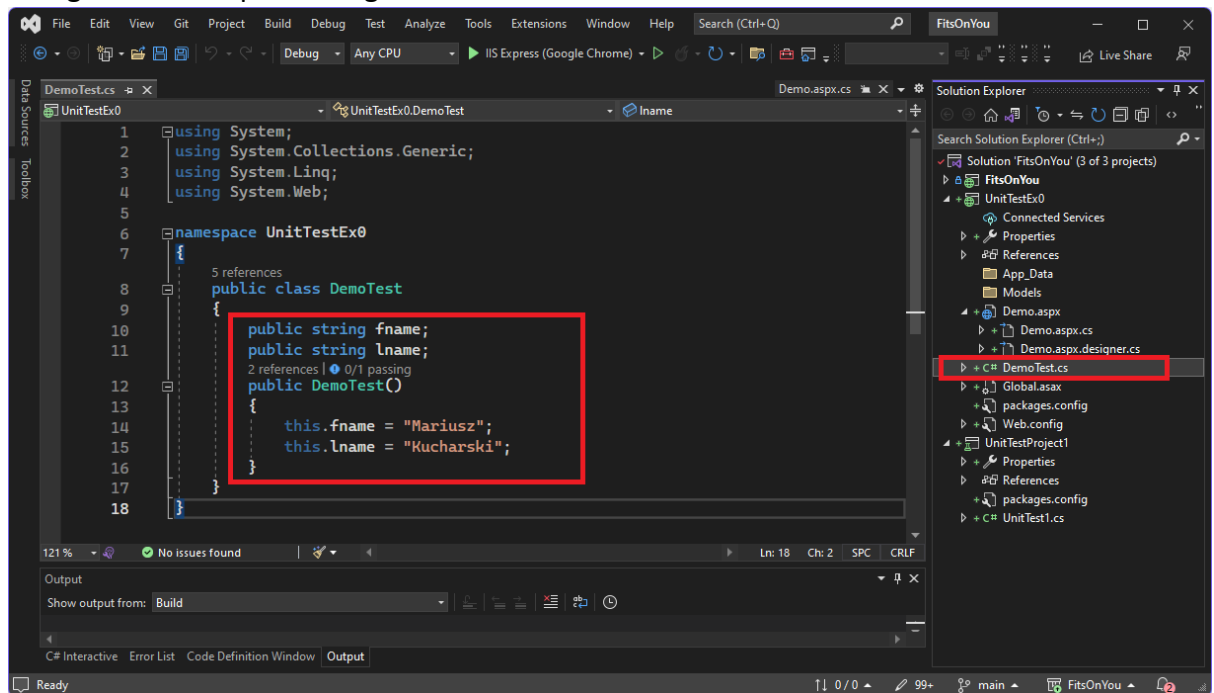
Now we need to pass a reference to our UnitTestEx0 project in UnitTestProject1.



Click the solution and now we can select the project that to which the references are to refer.

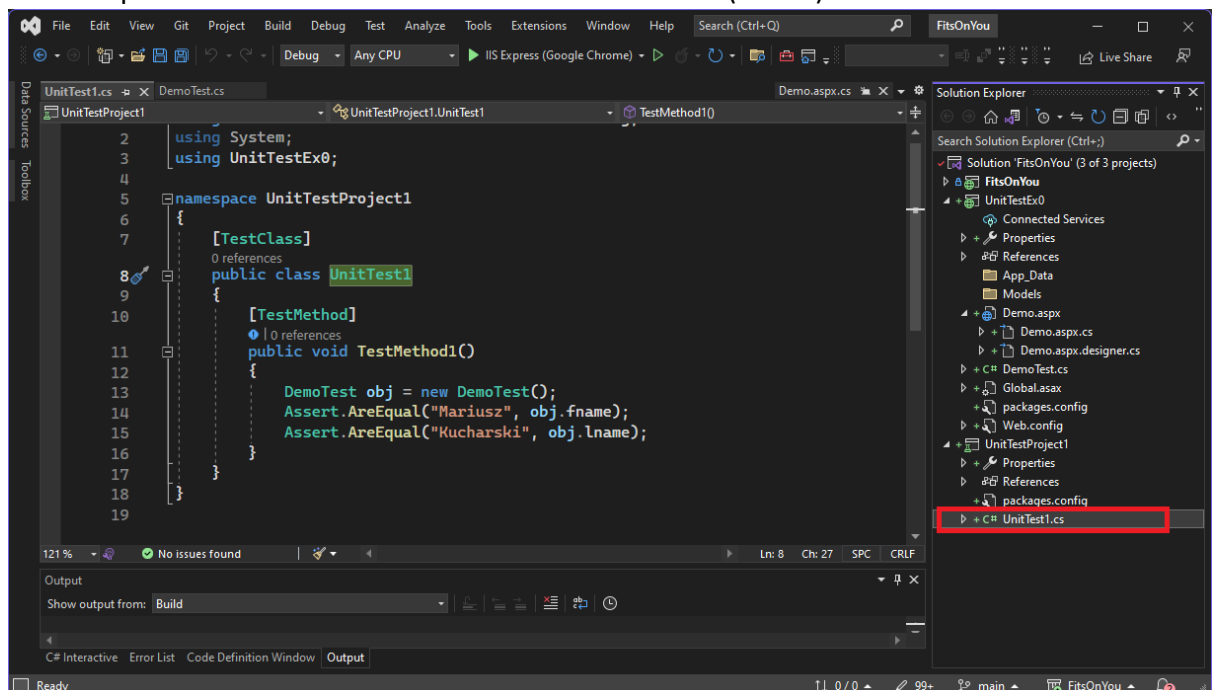


I want to test this class regarding values that should read fname as Mariusz and lname as Kucharski, so for this particular test purpose, we need to create a separate class that in our case it is a **DemoTest** class. In this **DemoTest** class (included in UnitTestEx0) I put two string variables representing first name and last name.



Every good project has a lot of tests that "assert" our assumptions. Every method you write should have a lot of test methods or "facts" that express how it should work. However, does it work how we expected? Let's check it out.

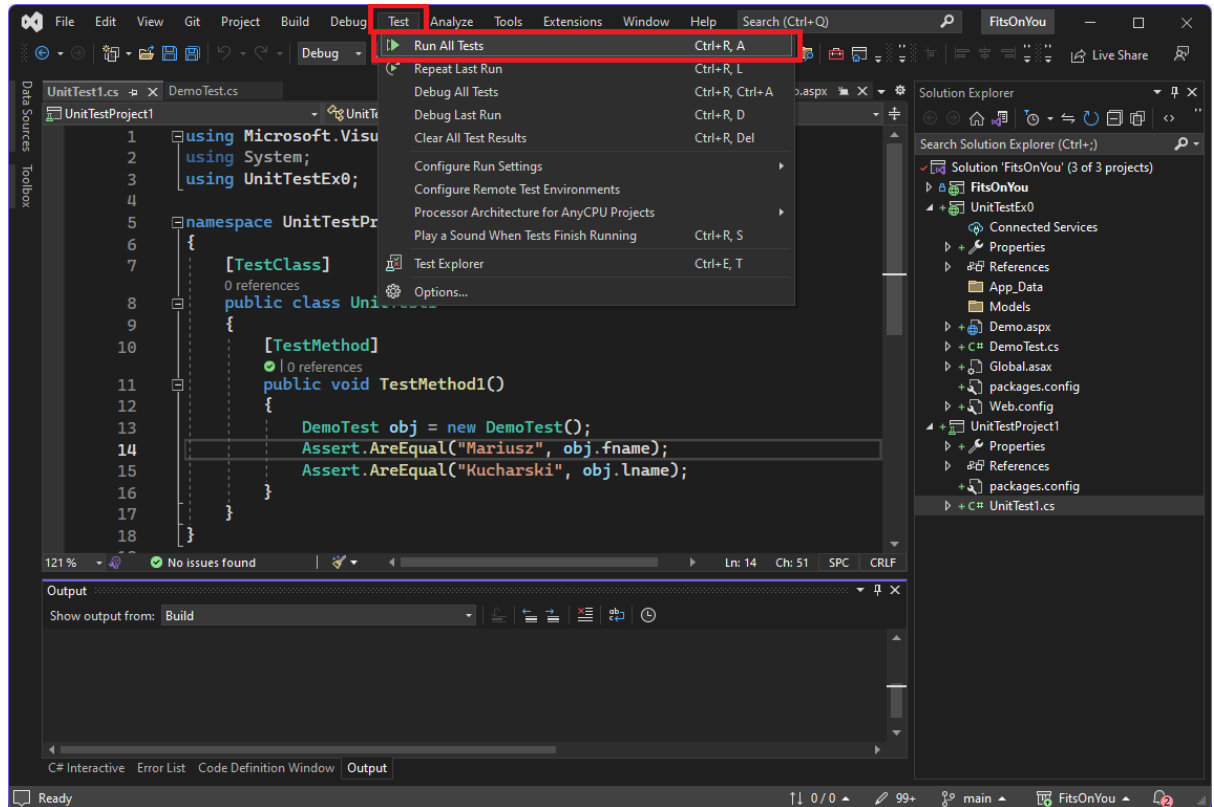
I wrote a simple method `UnitTest1k` class (in `UnitTestProject1` Project). This is the code we will use as a "testing tool". I want to test values like showed below using **Assert** method which is provided as a unit test framework that we use (**NUnit**).



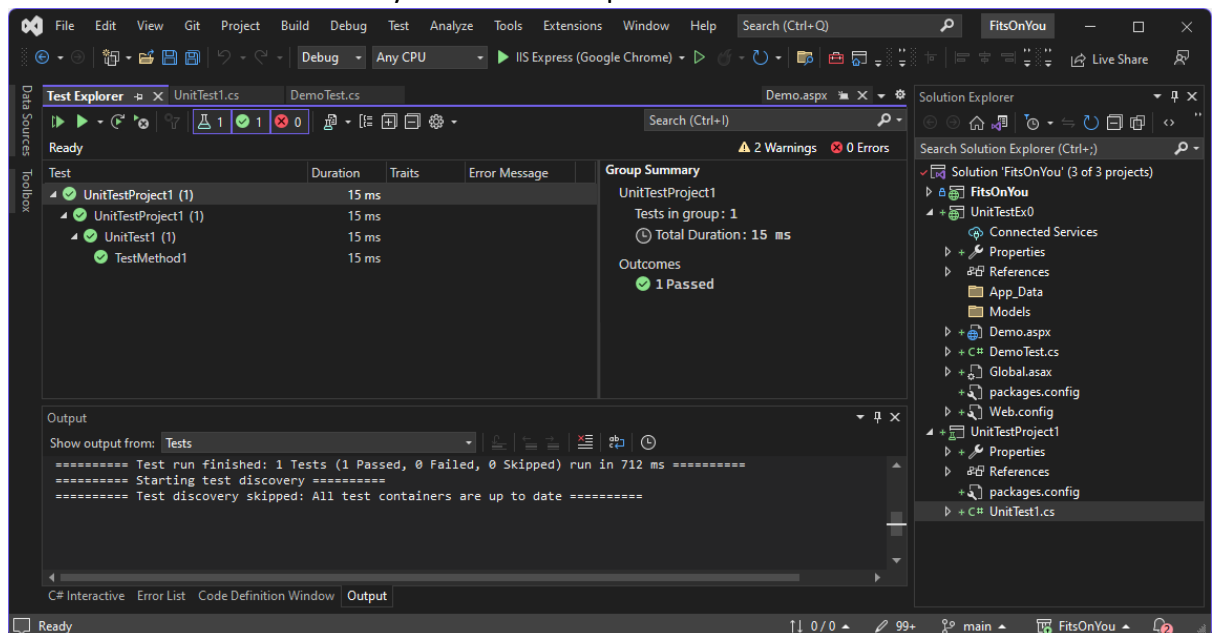
Assert have a method that is **AreEqual**. In AreEqual method we have to pass two values: **expected** (Mariusz) and **actual** (as an object fname). The same is for expected "Kucharski" and actual value that is object lname.

Let's test it

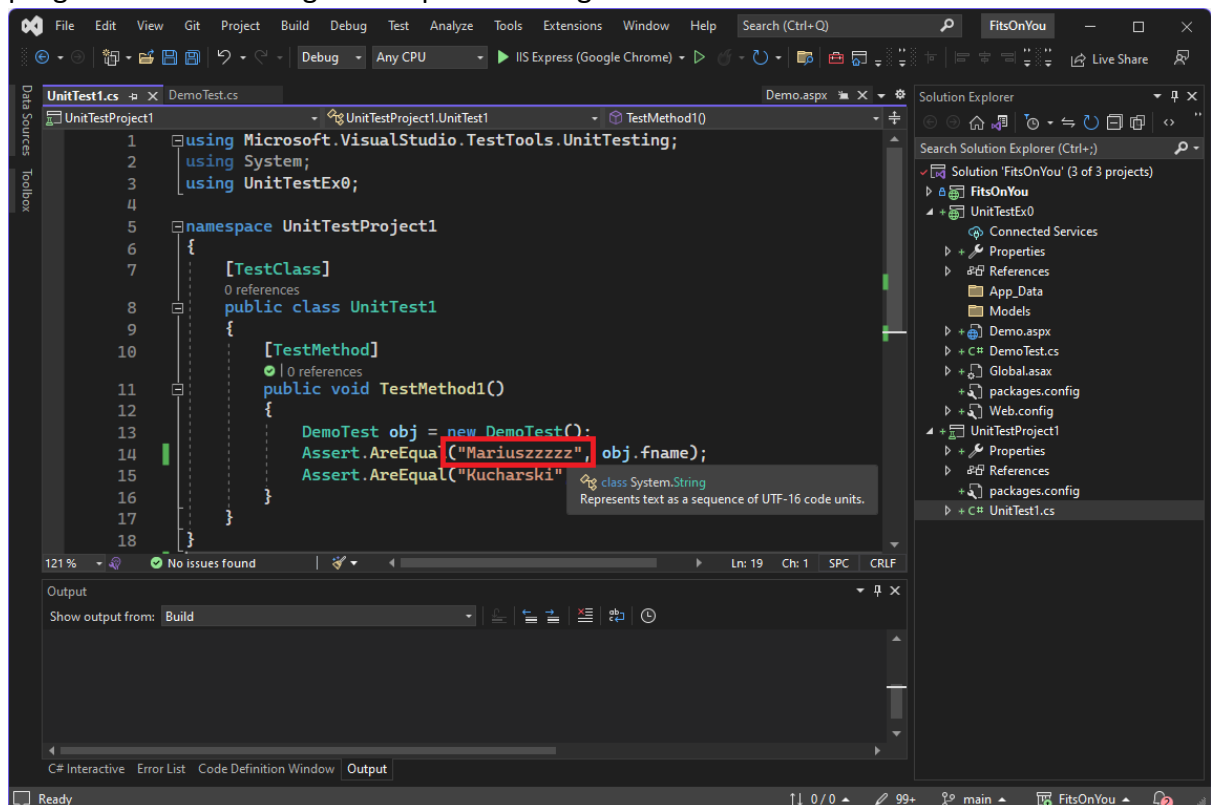
Click "Run All Tests" in the menu of Visual Studio



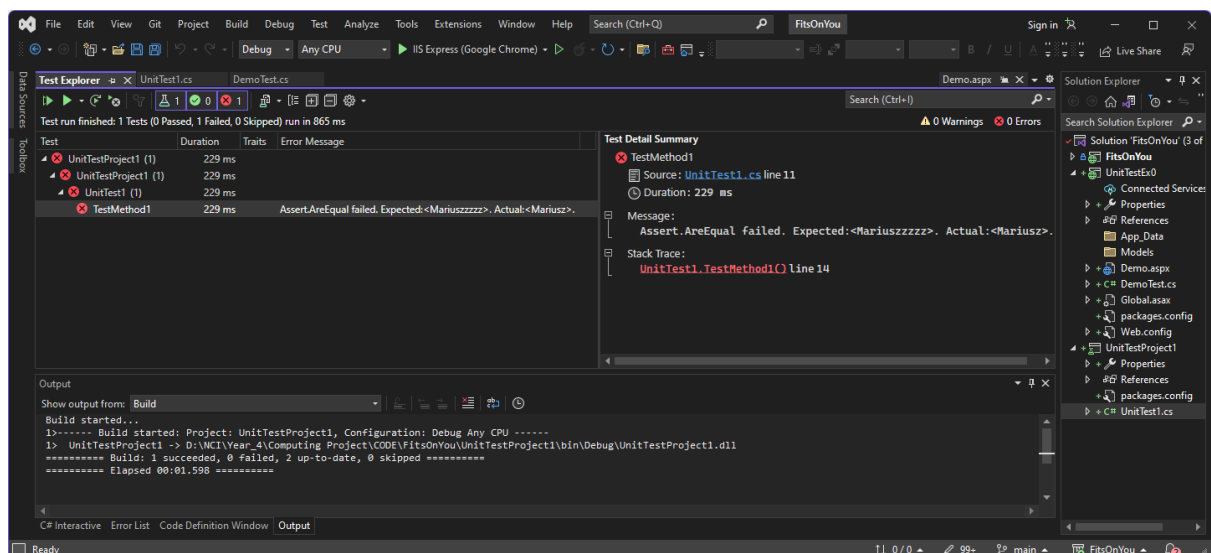
This is a result of this test. As you see the test passed. Test lasted for 15ms.



Now, let's break something and let's see if the test will show an error in the code of our program. We can change the expected string variable as below.



Result below shows error.



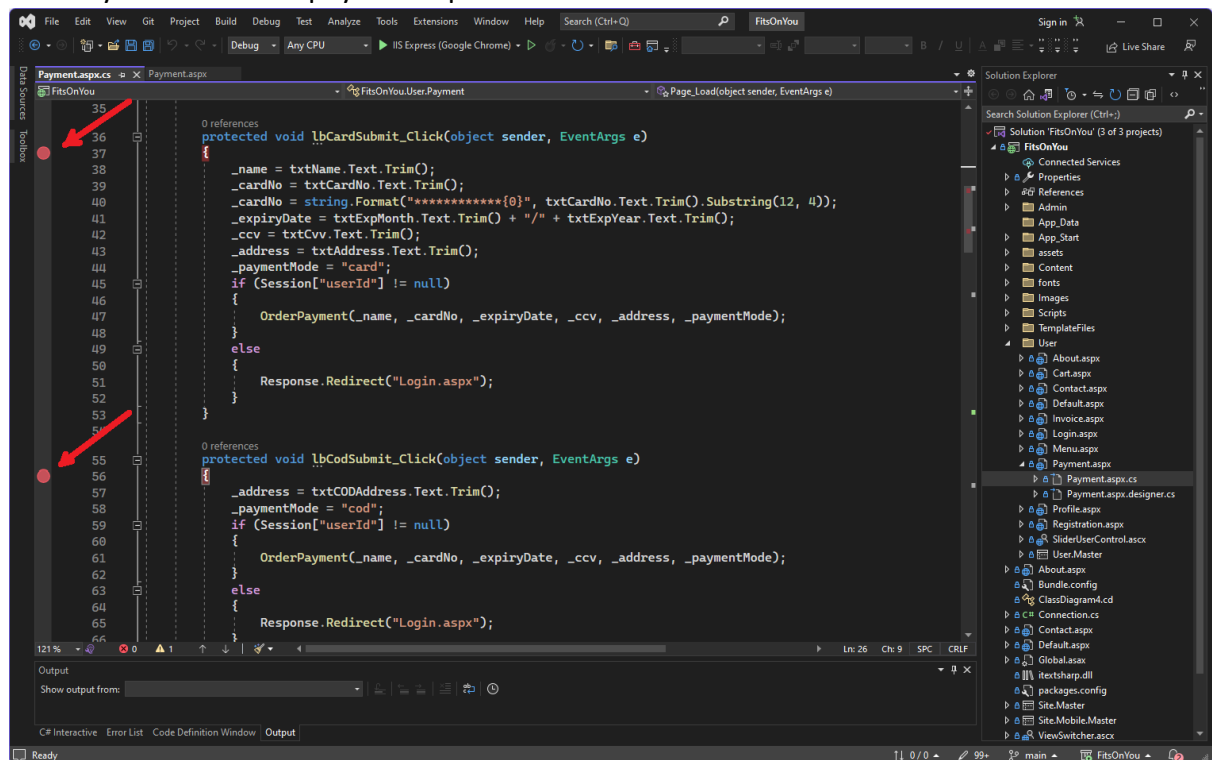
The test detected an error and a mismatch between the expected and actual values. As you can see, the expected value is not what it should be. We can determine exactly what type of error it is and where it is located.

This was just an example, however, as you can see, we can apply this kind of tests to each individual check and comparison of expected and actual values.

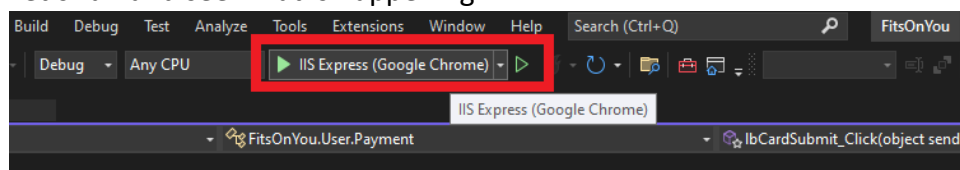
2.5.2. How I performed Integration Tests

As I mentioned earlier, integration tests verify the interaction between different components, modules, or layers in an application. In my project, I carried out these tests every time I integrated the program code with the database. I must say that they were even necessary for the correct operation of the application. I performed these tests manually using the debugging function, which in Visual Studio has a very extensive form.

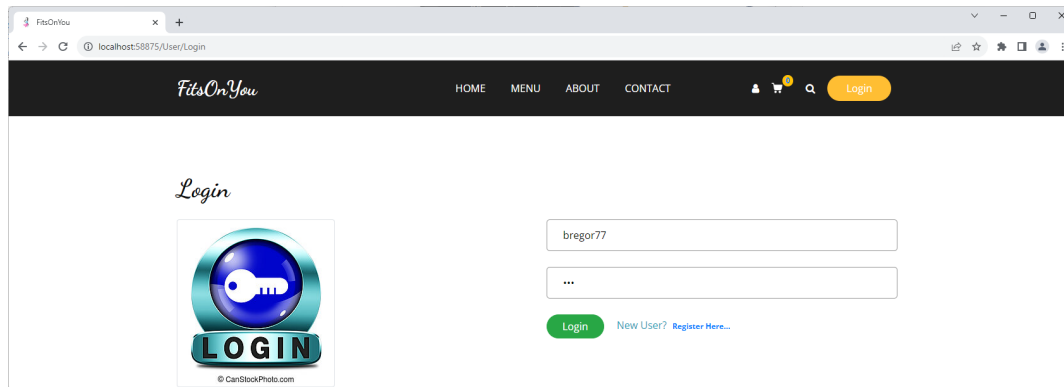
Below I will try to present an example of how I tested the Payment class in my project and how I checked the interaction of operations contained in the code with the database. Firstly I put a break point for the CardSubmit_Click and CodSubmit_Click. In this case, we will be able to track the payment approval using the Payment by Card and Cash on Delivery as the second payment option.



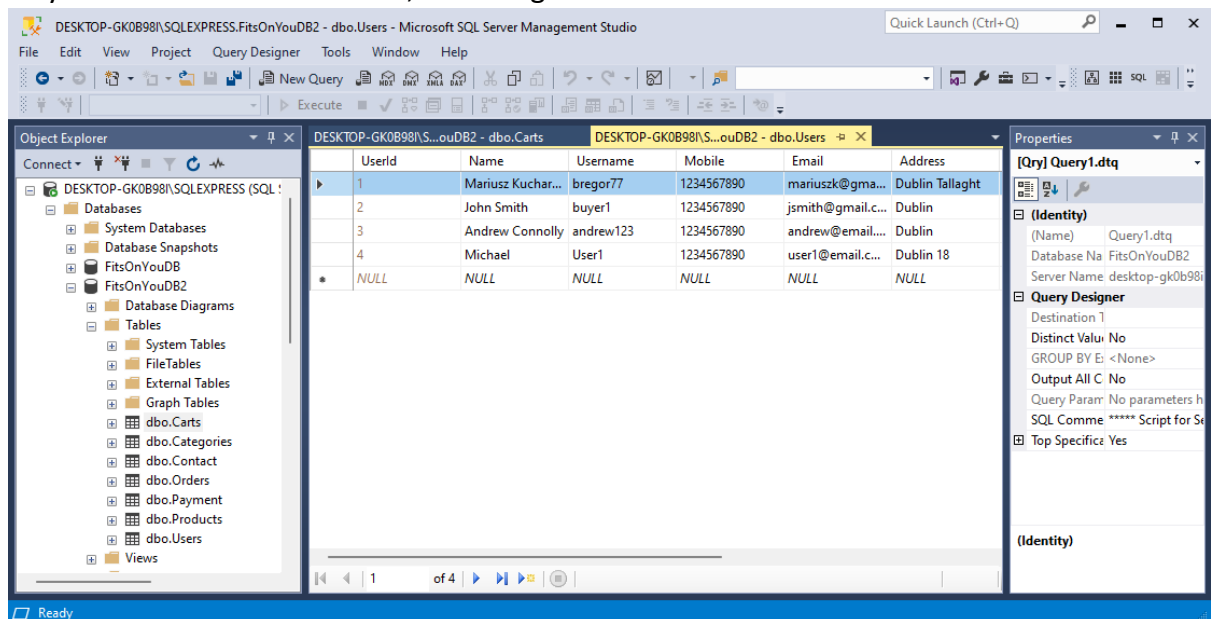
Let's run and see what is happening.



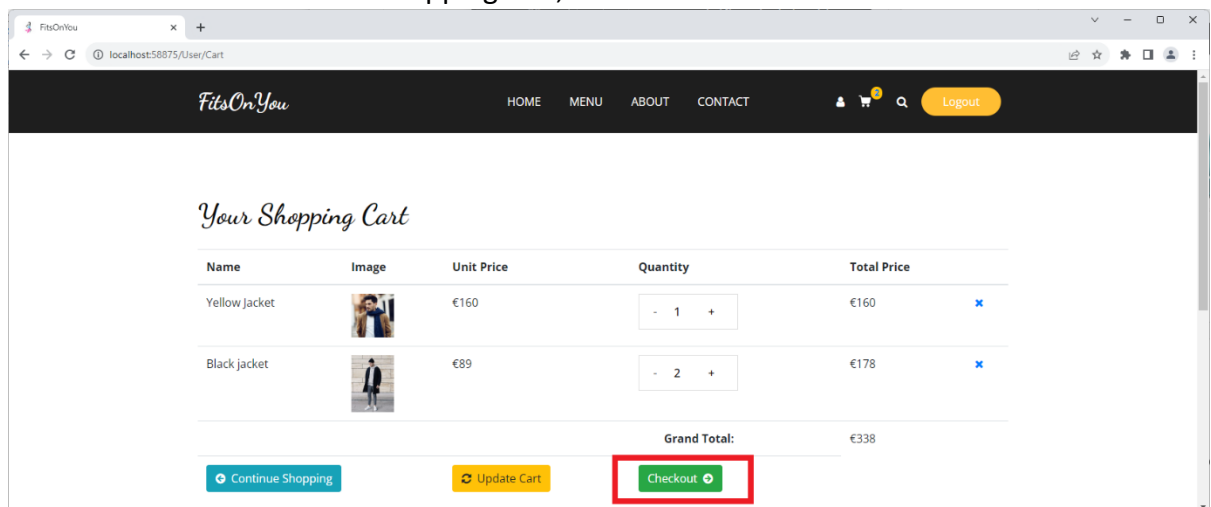
Application started, so let's login. I logged in as a user bregor77.



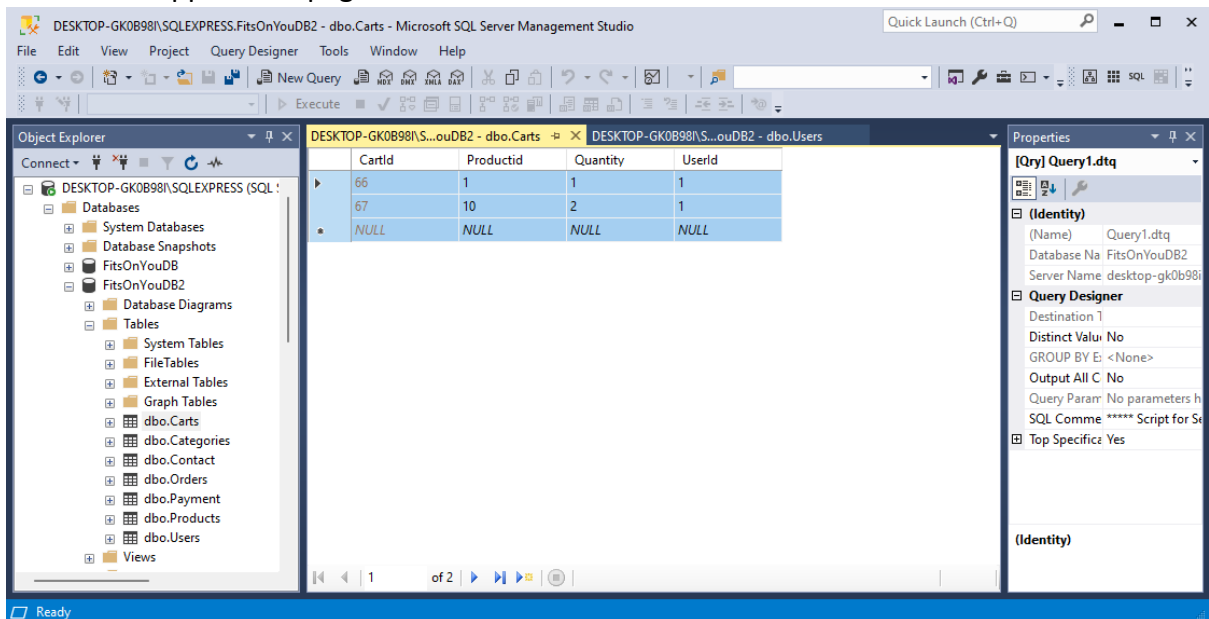
As you can see in the database, user bregor77 has number 1 in the UserId table.



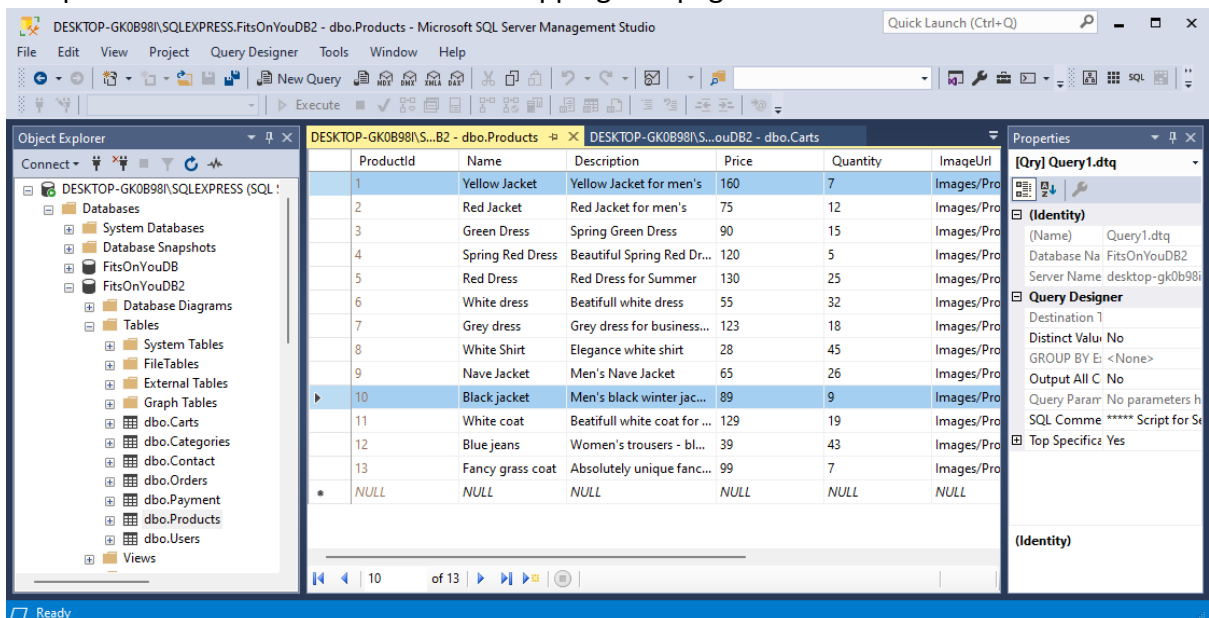
I added some 2 items to the shopping cart, so now we can click Checkout button.



Also, let's check in the database if these items has been added to the Cart table for UserId 1. ProductId 1 and ProductId 10 have been assigned to UserID 1. The quantity of products placed in the shopping cart table corresponds to the number of products added to the cart on the application page.



If we check what the ID number of these products means, it will turn out that these are exactly the products that we just added to the basket on the application page. The product and price match what we saw on the shopping cart page.



Let's come back to our app. We are redirected to the payment options page. As you can see, total payment matches the total payment value on the shopping cart page. Now, if you click Confirm Payment, you will see that form fields are required to be filled in.

The screenshot shows a web browser window with the URL `localhost:58875/User/Payment`. The page displays a payment form with two options: **Credit Card** (selected) and **Cash On Delivery**. The form fields are as follows:

- Card Owner ***: Input field with placeholder text "Card Owner Name".
- Card number ***: Input field with placeholder text "Valid card number" and a card icon.
- Expiration Date ****: Two input fields for "MM" and "YYYY".
- CWV ***: Input field with placeholder text "CWV No.".
- Delivery Address ***: Input field with placeholder text "Delivery Address".

A **Confirm Payment** button is located below the form fields. Below the button, the **Order Total: ₹ 338** is displayed. Below the total, the text *Fix the following errors* is shown, followed by a list of errors:

- Name is required
- Card number is required
- Expiry month is required
- Expiry year is required
- CWV no. is required
- Address is required

Let's fill in the form fields, click Confirm Payment once again and see what happens.

The screenshot shows a web browser window with the URL `localhost:58875/User/Payment`. The page title is "Order Payment". The form contains the following data:

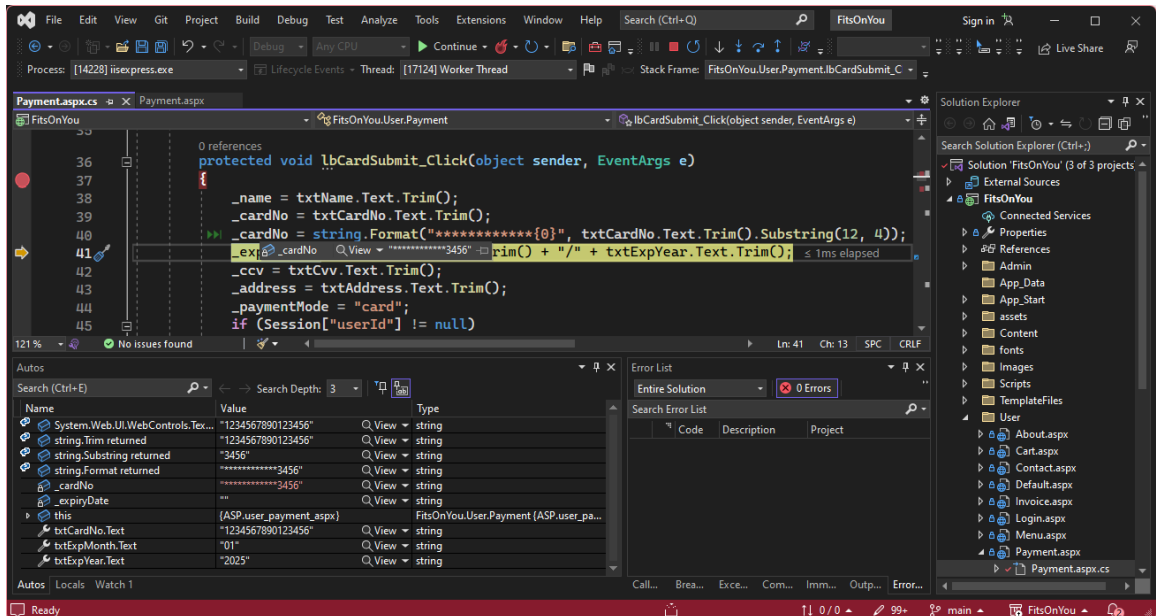
- Payment Mode: Credit Card, Cash On Delivery
- Card Owner: Mariusz Kucharski
- Card number: 1234567890123456
- Expiration Date: 01 / 2025
- CVV: 123
- Delivery Address: Dublin 24, Tallaght Square.
- Order Total: € 338

Now we can go back to the code. Press F10 and to go further and see all the steps of processing the program code and interacting with the database where all the information previously entered in the form by the user is stored. It should be mentioned that the manipulation of these data takes place both at the level of the stocking code and the previously mentioned Stored Procedures in the database.

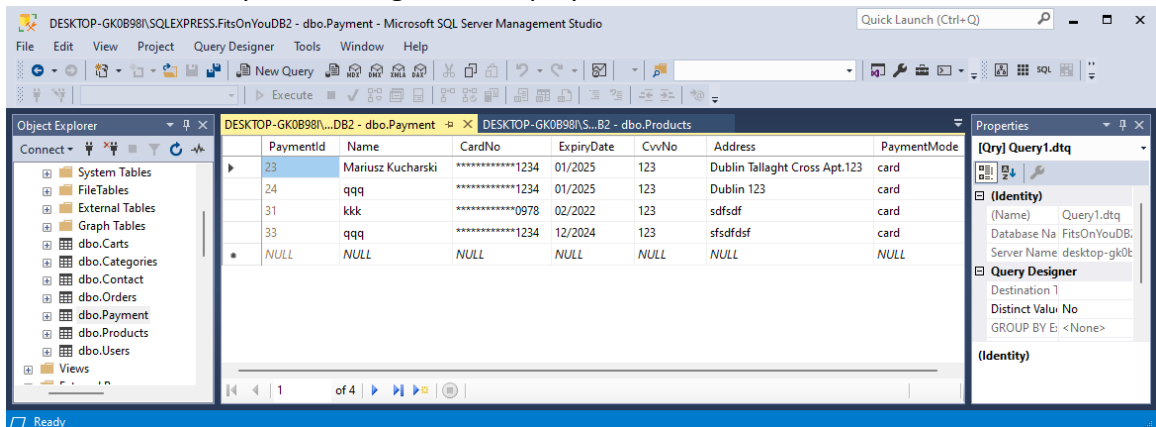
The debugger goes through all the parameters, such as the cardholder's name, card number, CVV number, address, type of payment. These are the parameters we saw when entering data in the form earlier.

```
36  
37  
38  
39  
40  
41  
42  
43  
44  
0 references  
protected void lbCardSubmit_Click(object sender, EventArgs e)  
{  
    _name = txtName.Text.Trim();  
    _cardNo = txtCardNo.Text.Trim();  
    _cardNo = string.Format("*****{0}", txtCardNo.Text.Trim().Substring(12, 4));  
    _expiryDate = txtExpMonth.Text.Trim() + "/" + txtExpYear.Text.Trim();  
    _ccv = txtCvv.Text.Trim();  
    _address = txtAddress.Text.Trim();  
    _paymentMode = "card";  
}
```

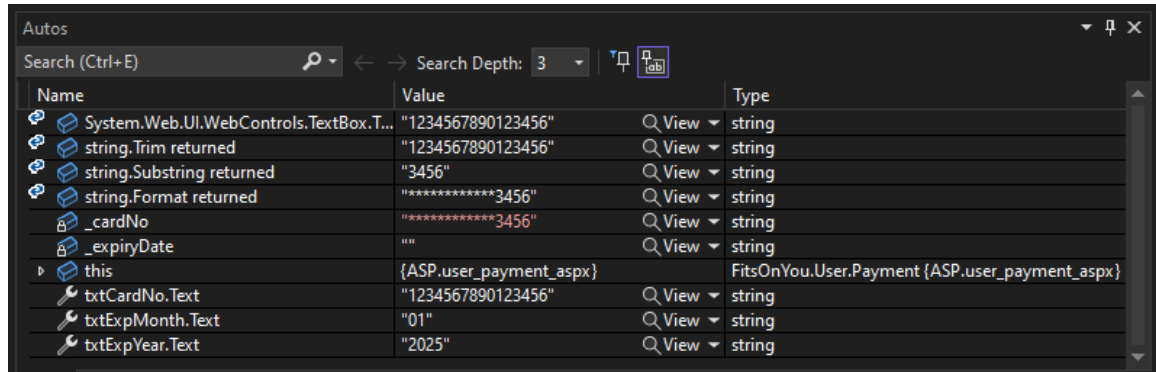
For example, here we can see how the information about the payment card number is displayed. It can be seen that the masking of the first 12 out of 16 digits on payment cards works here, and only the last 4 digits are uncovered. This is responsible for the appropriate formatting of this variable like that ("*****{0}") and displaying the value for this variable in the format (12, 4) where the first 12 digits are starred, that means the first 12 digits are uncovered only the last 4 digits are displayed as they are returned as Substring which is not starred.



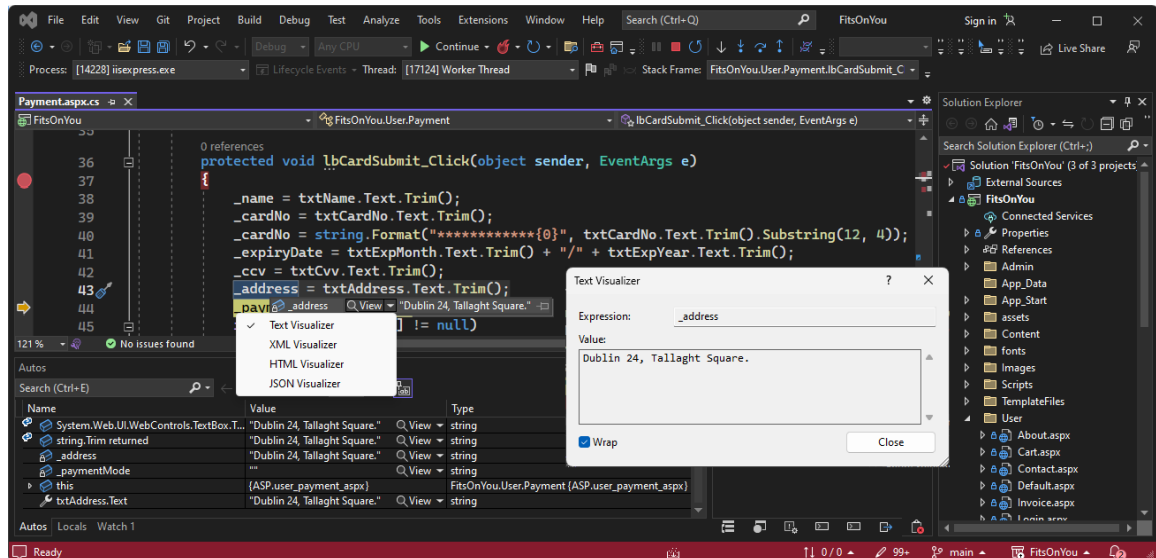
Of course, in the database in the payment table, the full values of payment cards are saved for the purpose of carrying out bank payment operations, but for security reasons and maintaining the conventions of storing this type of data, even for the system administrator, only the last digits are displayed.



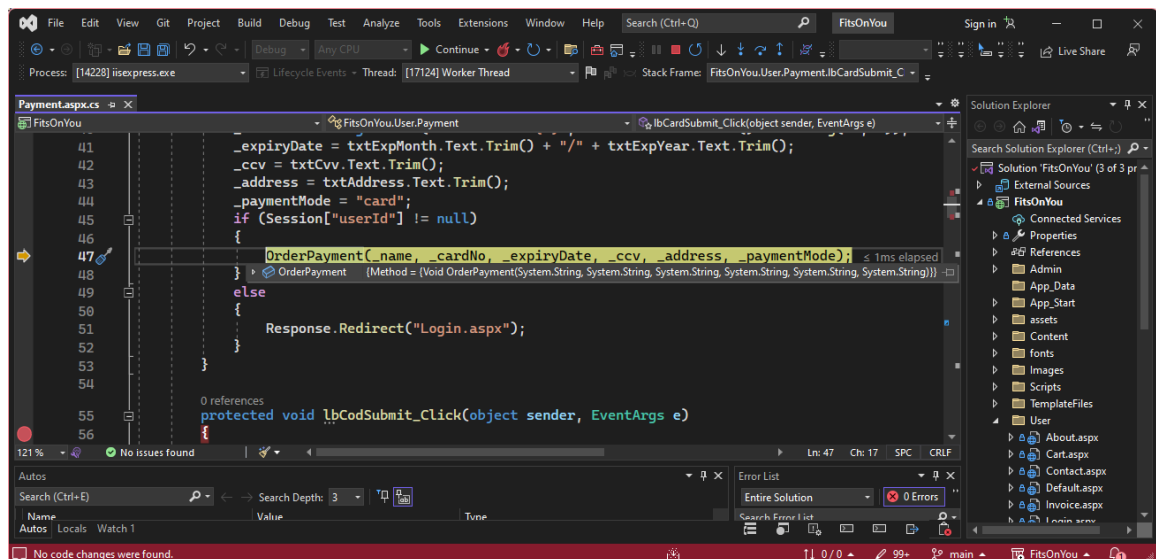
Below we can see in the event viewer what it looks like from the application code side.



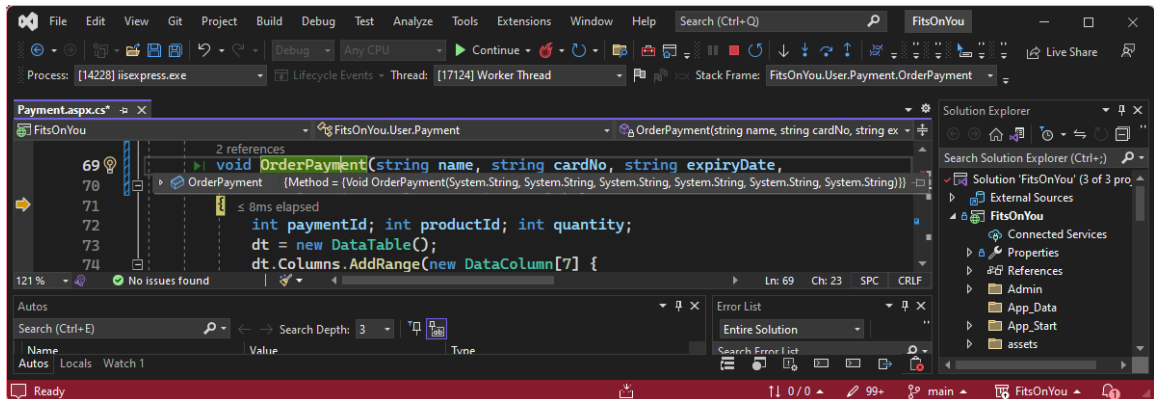
The same type of data review applies to viewing and analysing all previously entered information in the payment form (in this case by card). For example, here we see address information. These are exactly the data that we previously entered in the form. Visual Studio even allows you to visualize a preview of this data as plain text, XML, HTML and JSON. For example below we see this for the address parameter.



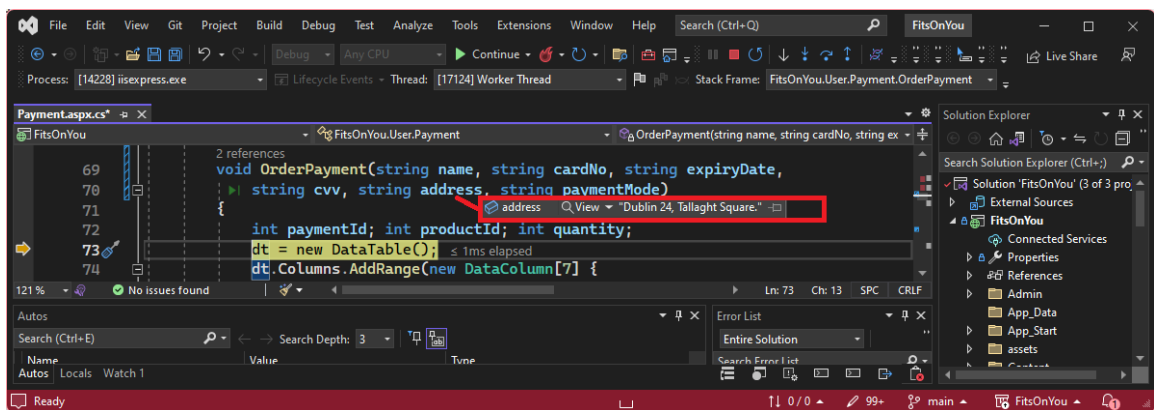
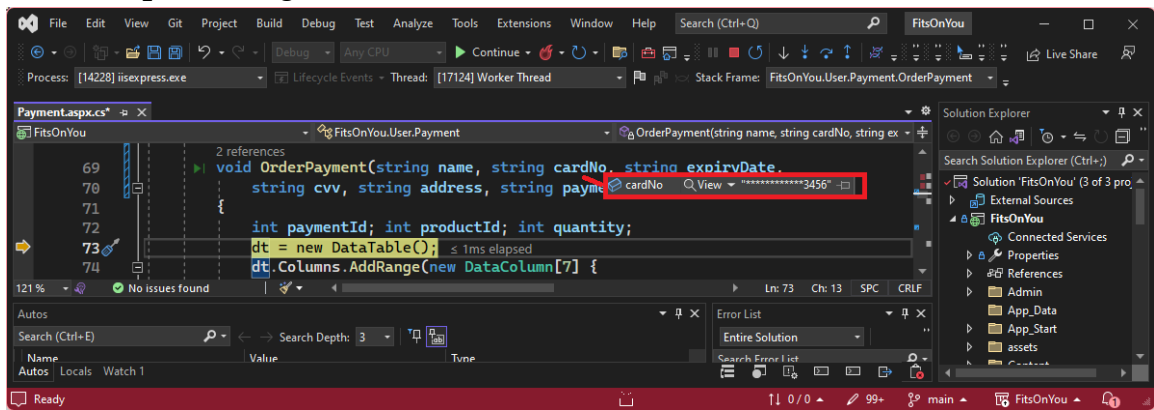
Now that we have reached the `OrderPayment` function, if we press F11 we will be taken to inside this function.



Here we can continue our investigation by checking the correct operation of the code and searching for possible errors by using F10 key.



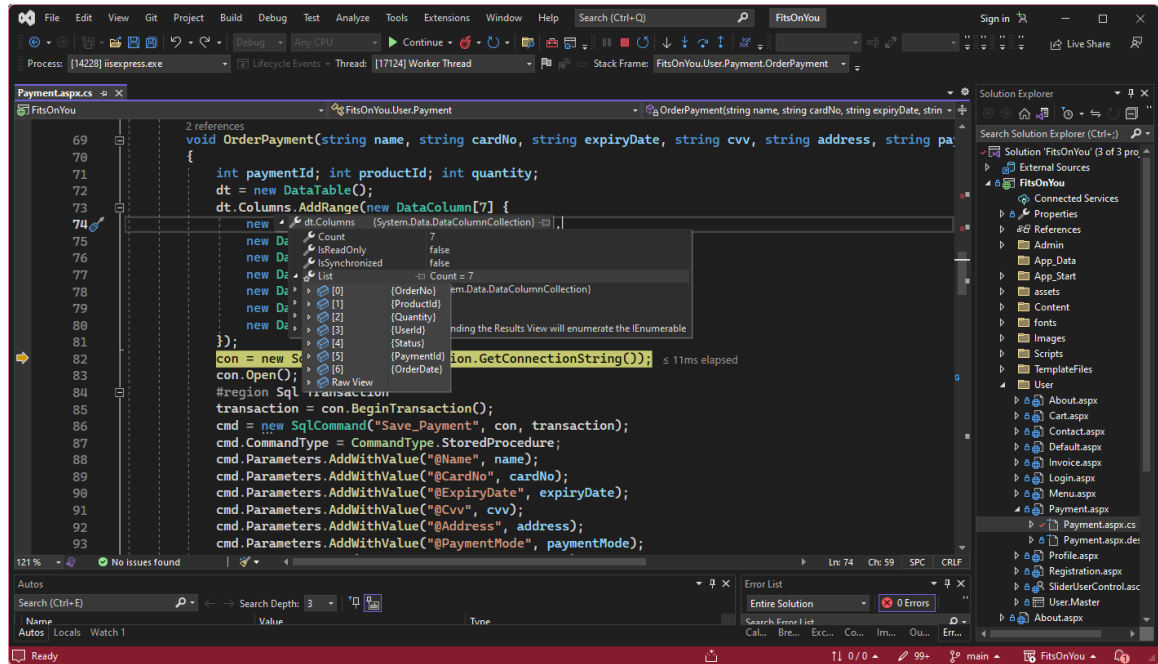
Here we can check if all form fields are correctly placed as parameters in the method `OrderPayment` e.g., for card number and address as shown below.



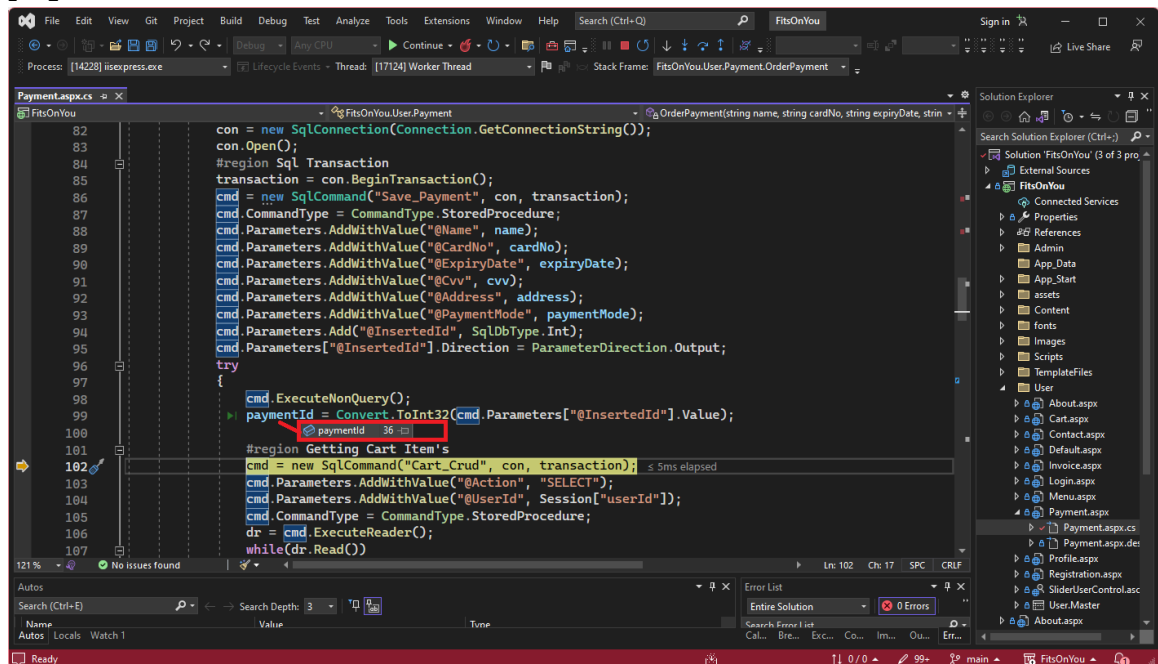
Now the columns are created.

```
dt = new DataTable();
dt.Columns.AddRange(new DataColumn[7] {
    new DataColumn("OrderNo", typeof(string)),
    new DataColumn("ProductId", typeof(int)),
    new DataColumn("Quantity", typeof(int)),
    new DataColumn("UserId", typeof(int)),
    new DataColumn("Status", typeof(string)),
    new DataColumn("PaymentId", typeof(int)),
    new DataColumn("OrderDate", typeof(DateTime)),
});
```

As you can see, 7 columns are created. If we expand the List tab, we can see exactly what columns are being created at the moment e.g., OrderNo, ProductId and so on.



Now, connection is open, transaction begin. We get here the payment ID that is paymentId number 36.



Now we need some data of that user in his shopping cart. Based on that we need to find the product ID and quantity of these items.

```
productid = (int)dr["ProductId"];
quantity = (int)dr["Quantity"];
```

productid=1

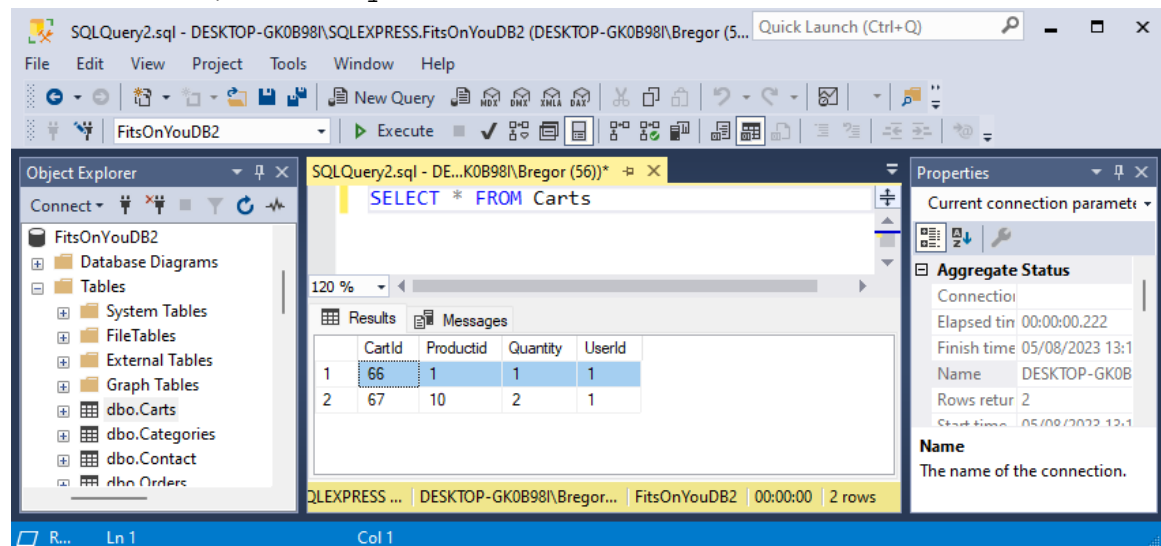
```
109
110 ▶ productId = (int)dr["ProductId"];
111     quantity = (int)dr["Quantity"]; ≤ 5ms elapsed
112     // Updated Product Quantity
113     UpdateQuantity(productId, quantity, transaction, con);
114     // Updated Product Quantity End
```

quantity=1

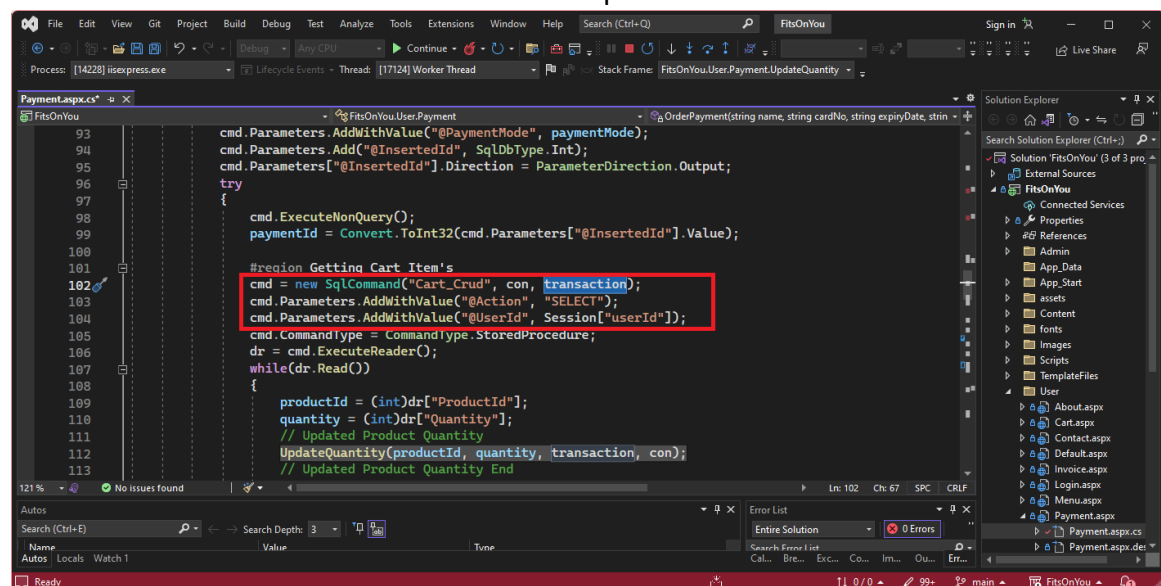
```
109
110 productId = (int)dr["ProductId"];
111 quantity = (int)dr["Quantity"];
112 // quantity 1 -> ct Quantity
113 UpdateQuantity(productId, quantity, transaction, con); ≤ 1ms elapsed
114 // Updated Product Quantity End
```

Let's compare it with the tables in database. Everything is correct.

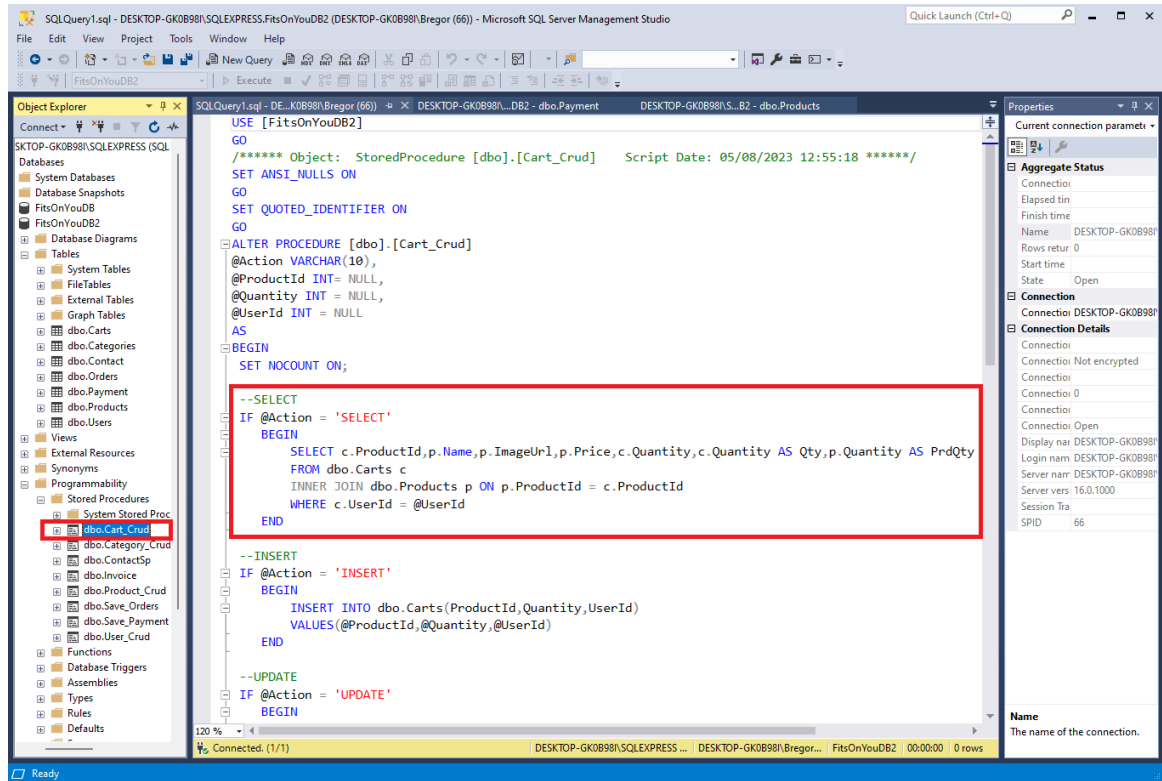
ProductId=1, Quantity=1



We initiate the **Cart_Crud** stored procedure created in our database for **@Action SELECT** where we select the indicated tables for the specified **UserId**.



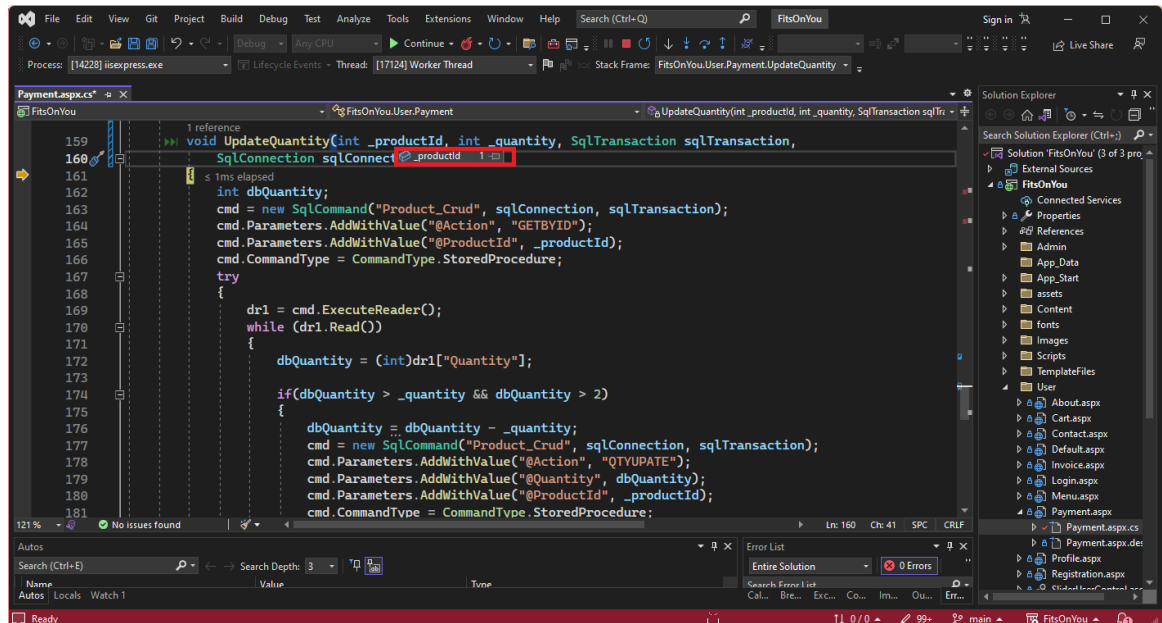
Below we see what it looks like from the database side.



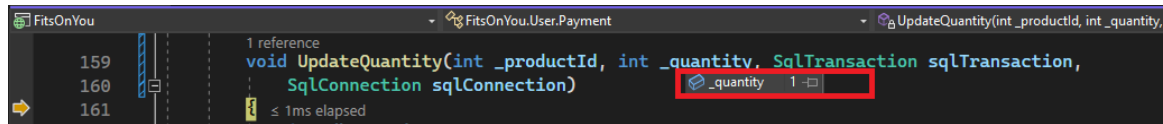
Let's go back to our code. In this moment of our debugging, if we press F11, we will be taken to inside function UpdateQuantity.



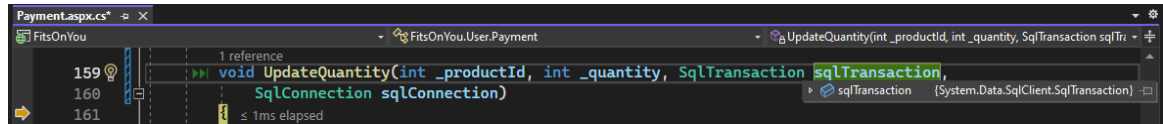
As you can see, productId is 1.



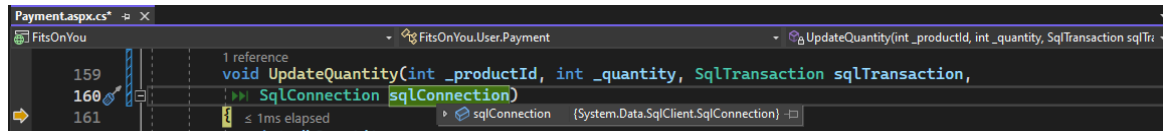
quantity also is 1



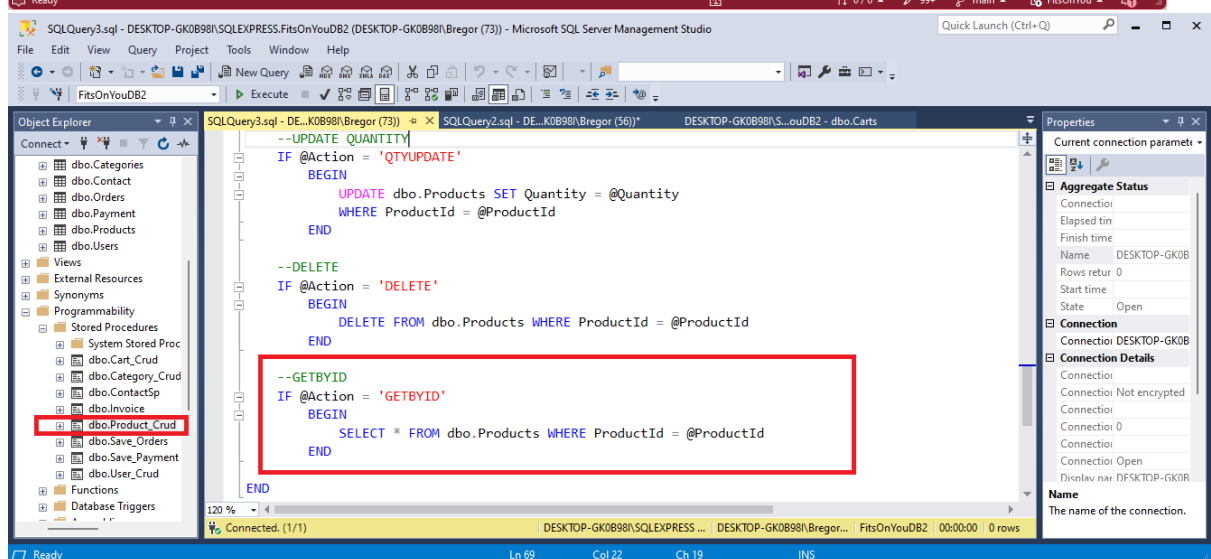
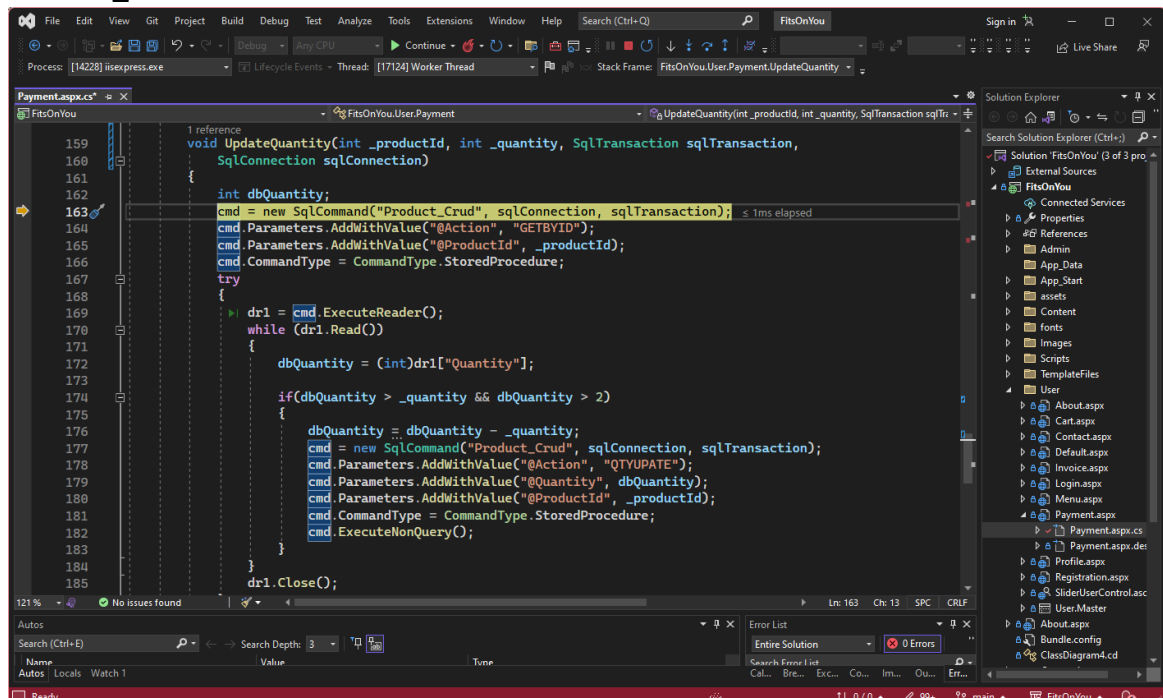
sqlTransaction is active



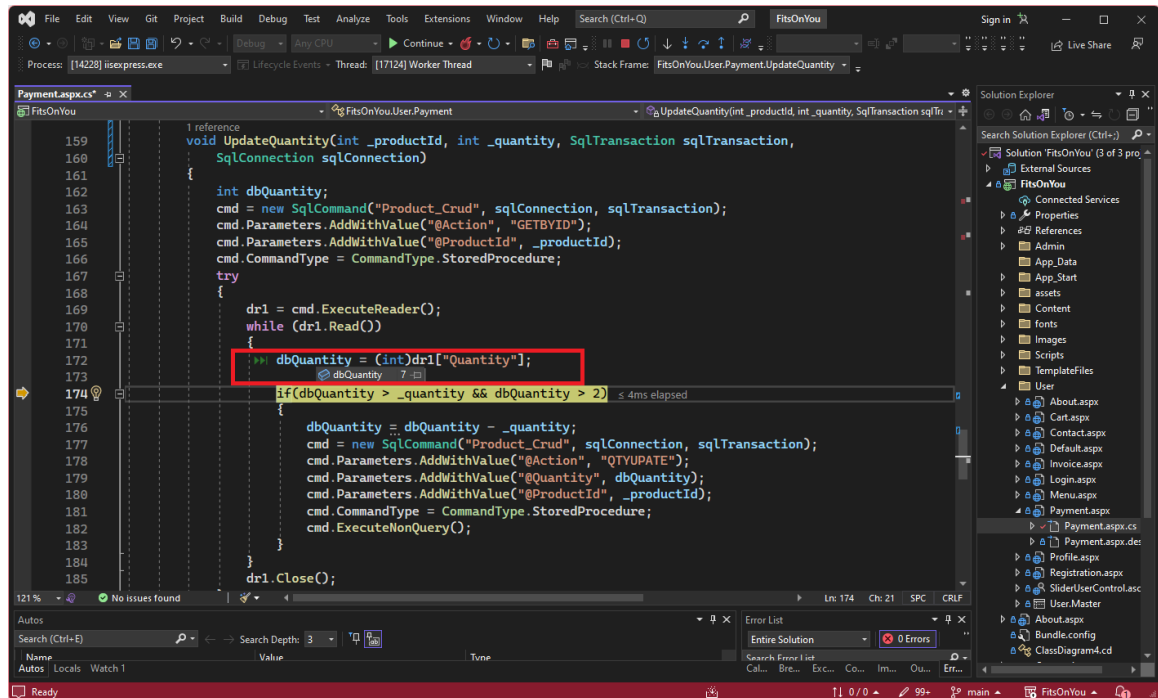
sqlConnection is opened



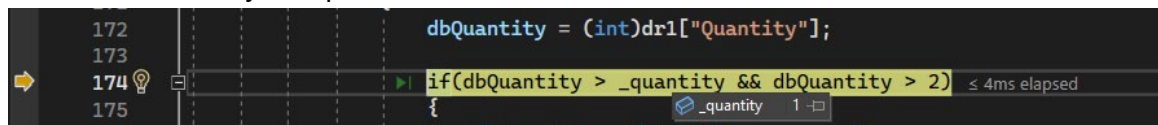
Now we initiate another @Action GETBYID for ProductId performed in Stored Procedure Product_Crud.



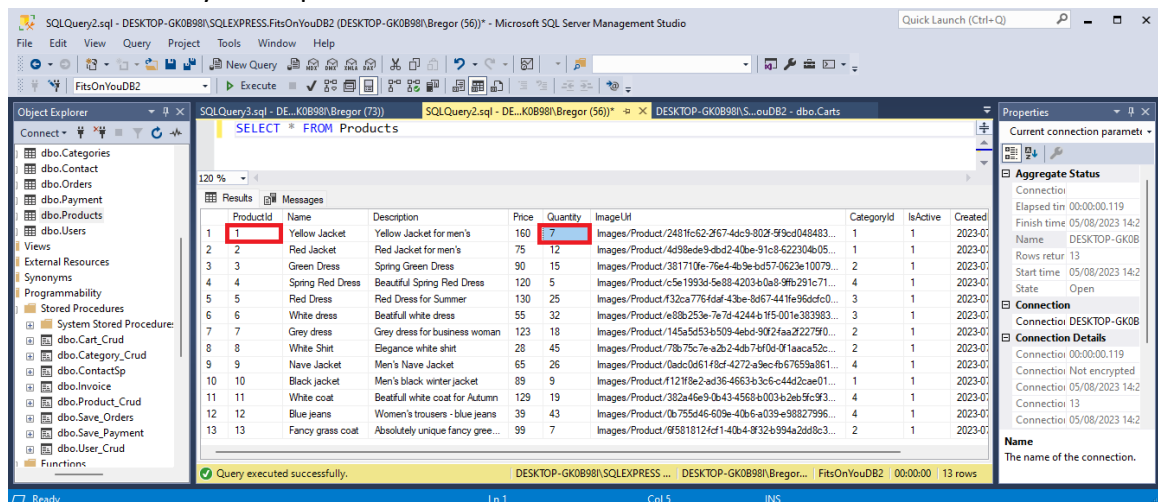
We need to check how much of the product is in stock, so that the transaction is possible, as we cannot sell a product that has zero stock. We can see that in database there is 7 pieces of this product.



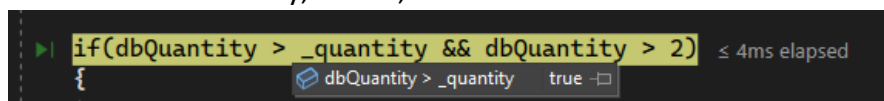
We want to order just 1 product



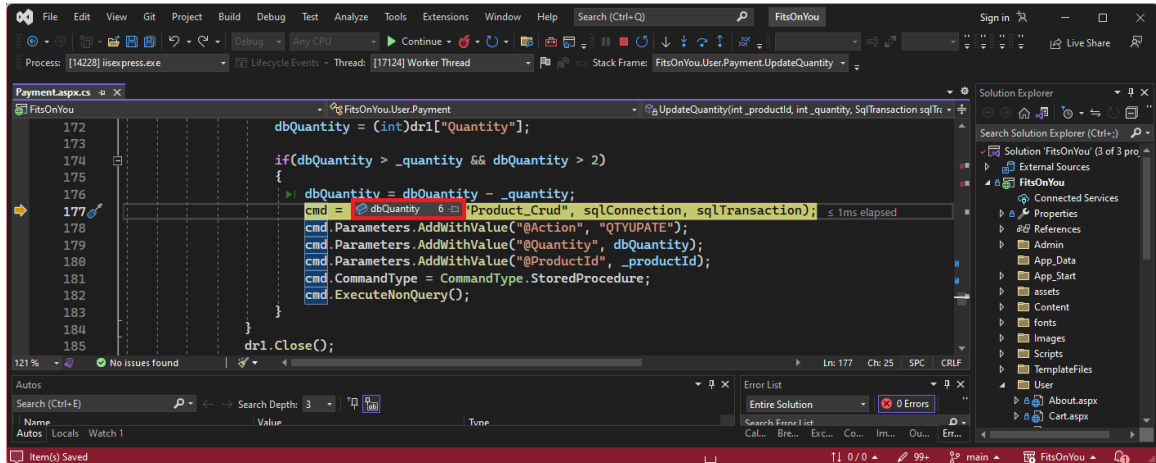
Just to confirm, let's take a look at what it looks like in our database. Quantity here also shown Quantity=7 for product with ID1.



Thus, if this statement, which represents the quantity of products in stock and products that user wants to buy, is true, the transaction is feasible.



Here we see that the quantity has been updated. As you can see, the number of products in the database has decreased by the amount the user wants to buy (7-1=6).



Now we need to update this quantity in database in the product table for productId via initiate @Action QTYUPDATE in Product_Crud for quantity of selected product ID.

dbQuantity now should equals 6

```

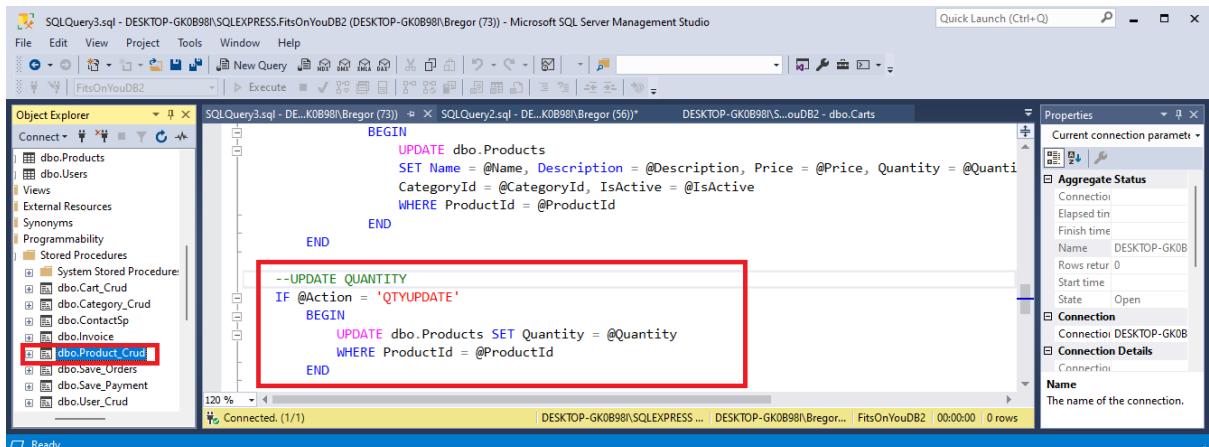
dbQuantity = dbQuantity - _quantity;
cmd = new SqlCommand("Product_Crud", sqlConnection, sqlTransaction);
cmd.Parameters.AddWithValue("@Action", "QTYUPDATE");
cmd.Parameters.AddWithValue("@Quantity", dbQuantity);
cmd.Parameters.AddWithValue("@ProductId", _productId);
cmd.CommandType = CommandType.StoredProcedure;
cmd.ExecuteNonQuery();
    
```

for productId number 1

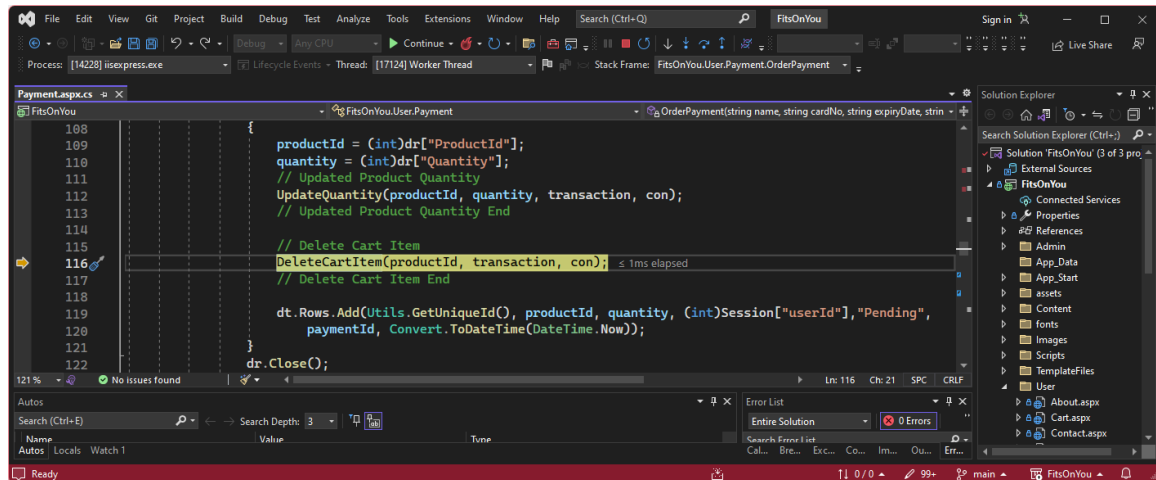
```

dbQuantity = dbQuantity - _quantity;
cmd = new SqlCommand("Product_Crud", sqlConnection, sqlTransaction);
cmd.Parameters.AddWithValue("@Action", "QTYUPDATE");
cmd.Parameters.AddWithValue("@Quantity", dbQuantity);
cmd.Parameters.AddWithValue("@ProductId", _productId);
cmd.CommandType = CommandType.StoredProcedure;
cmd.ExecuteNonQuery();
    
```

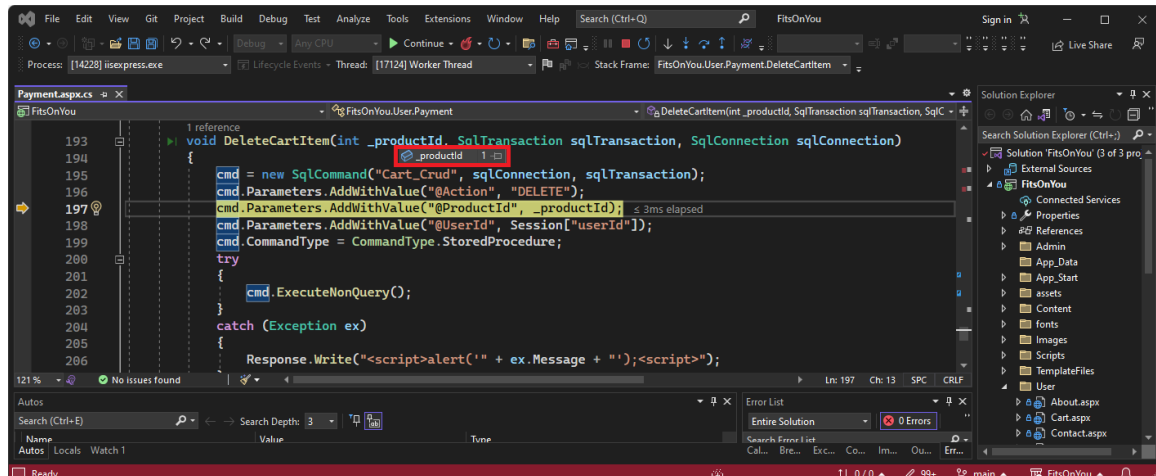
Below you can see how looks the @Action update quantity in Product_Crud in database.



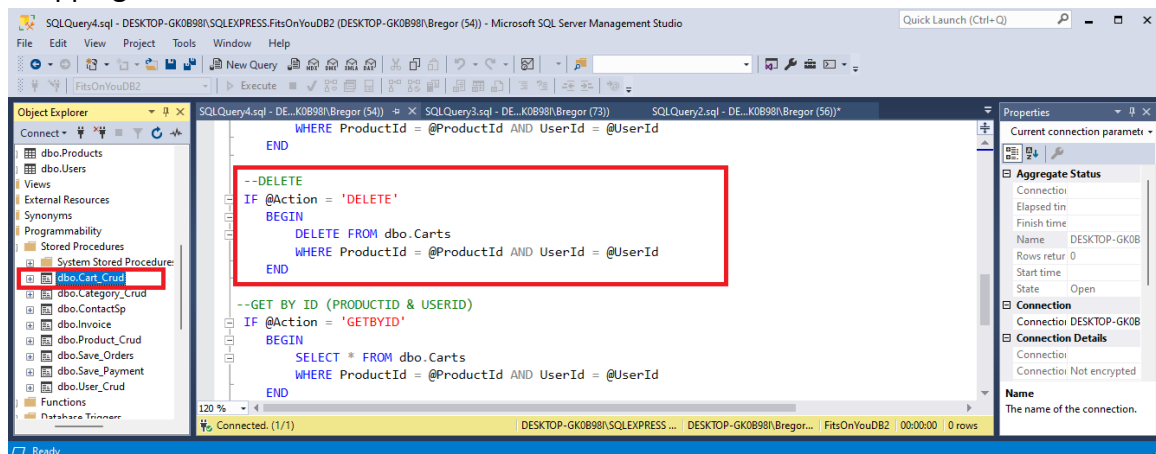
The update perform is closed when it is everything done successfully.
So, when **UpdateQuantity** is closed we can go do **DeleteCartItem**



At this stage of debugging press F11 and check what does this function look like inside.
We are moved to this DeleteCartItem method.

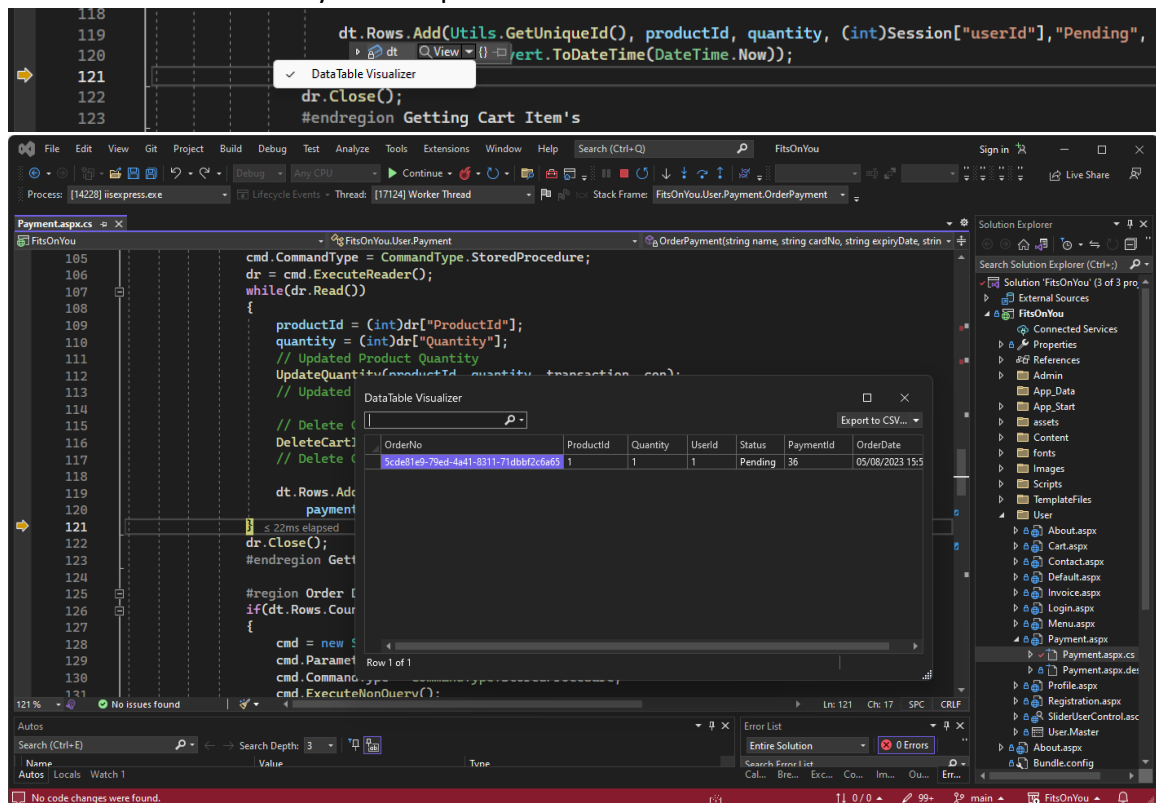


We want to delete productId 1 which is in the cart table. To do that we can call the **@Action DELETE** in the **Cart_Crud**, so we parse which product ID we want to delete in the shopping cart for which user ID.



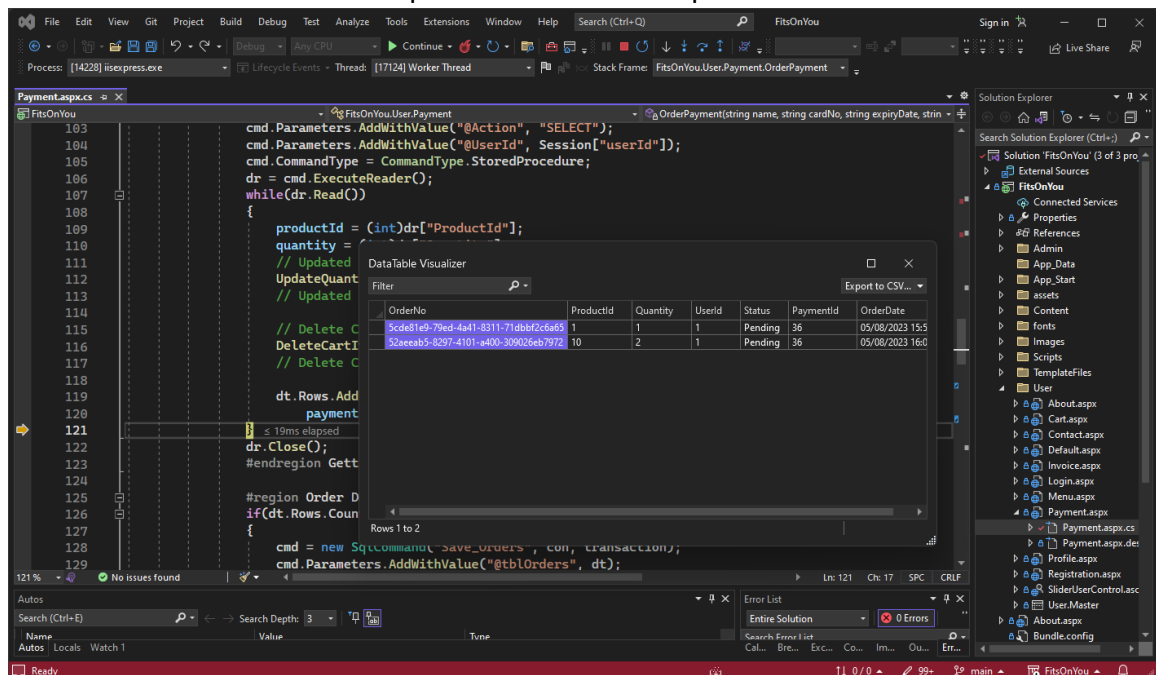
Thus, if update and delete are performed successfully, remains one more thing.

If in this section you can use visualize how the transaction looks so far and you can notice that we made order only for one productid.

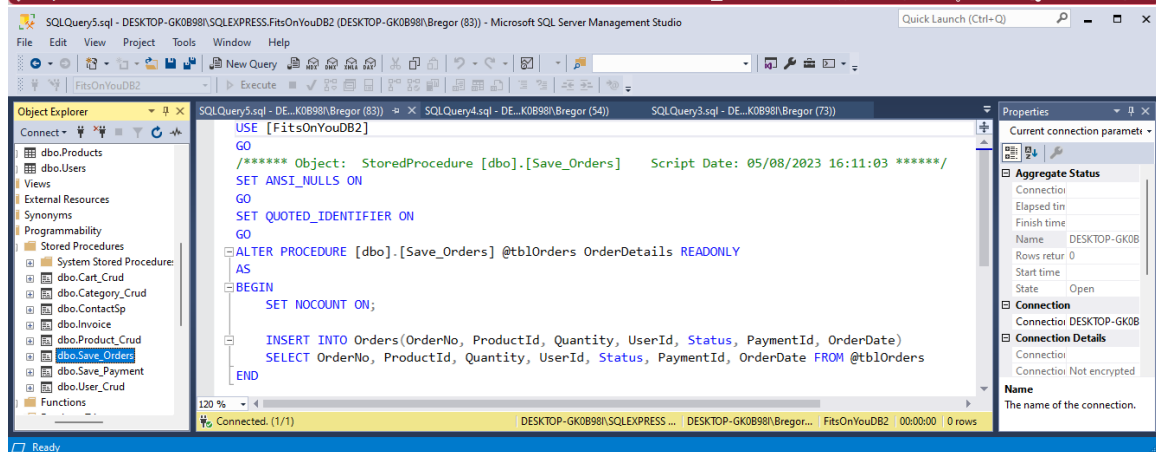
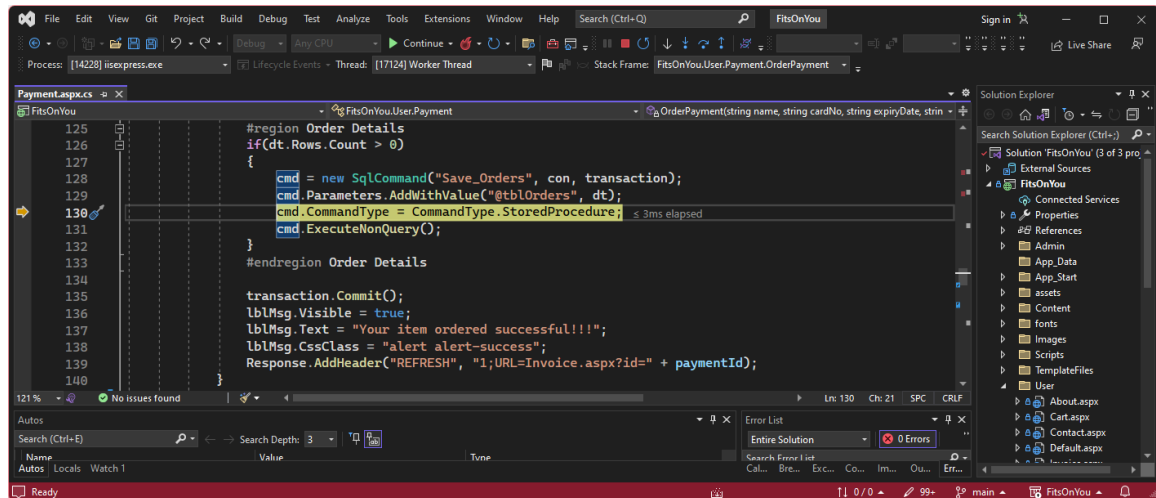


So, if we continue now, we'll notice that debugging takes us back through the loop "while" through the entire process we saw for the first product.

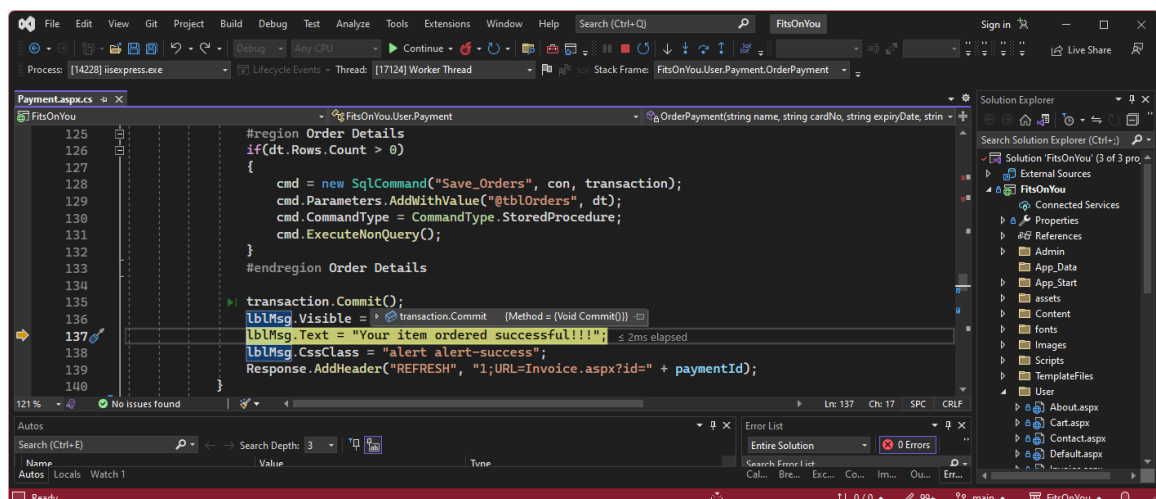
When we finish the second loop we will see the two products that ordered user.



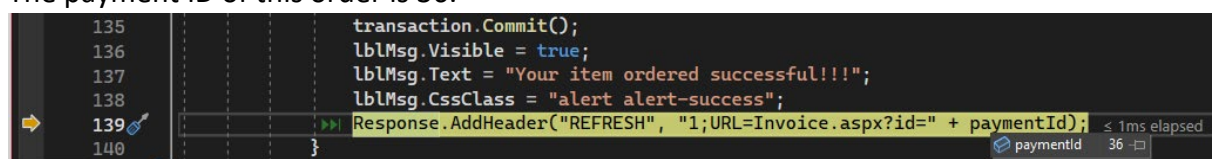
Now we need to pass all the data transaction and save the order in the database in the Orders table by call Stored Procedures **Save_Orders** and pass all the data transaction.



Transaction commit.



The payment ID of this order is 36.



When the connection is closed and if we press “Continue” at top of the Visual Studio, in this moment user should see the message that the item has been ordered successfully.

```

135         transaction.Commit();
136         lblMsg.Visible = true;
137         lblMsg.Text = "Your item ordered successful!!!";
138         lblMsg.CssClass = "alert alert-success";
139         Response.AddHeader("REFRESH", "1;URL=Invoice.aspx?id=" + paymentId);
140     }
141     catch (Exception)
142     {
143         try
144         {
145             transaction.Rollback();
146         }
147         catch (Exception ex)
148         {
149             Response.Write("<script>alert('"+ ex.Message + "');</script>");
150         }
151     }
152     #endregion Sql Transaction
153     finally
154     {
155         con.Close();
156     }
157 }

```

We can check this order via Purchased History in User Profile Information.

User Information

Mariusz Kucharski
bregor77
mariuszk@gmail.com
23/07/2023 08:50:50

[Edit Details](#)

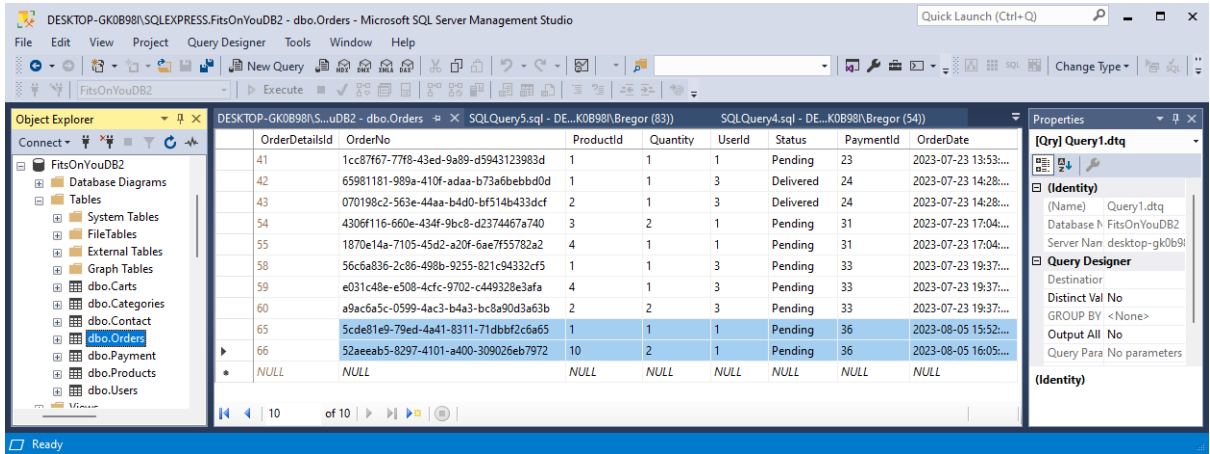
[Basic Info](#) [Purchased History](#)

1 Payment Mode: CARD		Card No: *****1234		Invoice	
Product Name	Unit Price	Qty	Total Price	OrderId	Status
Yellow Jacket	€160	1	€160	1cc87f67-77f8-43ed-9a89-d5943123983d	Pending
			€160		

2 Payment Mode: CARD		Card No: *****0978		Invoice	
Product Name	Unit Price	Qty	Total Price	OrderId	Status
Green Dress	€90	2	€180	4306f116-660e-434f-9bc8-d2374467a740	Pending
Spring Red Dress	€120	1	€120	1870e14a-7105-45d2-a20f-6ae7f55782a2	Pending
			€300		

3 Payment Mode: CARD		Card No: *****3456		Invoice	
Product Name	Unit Price	Qty	Total Price	OrderId	Status
Yellow Jacket	€160	1	€160	5cde81e9-79ed-4a41-8311-71dbbf2c6a65	Pending
Black jacket	€89	2	€178	52aeeab5-8297-4101-a400-309026eb7972	Pending
			€338		

Finally, we can check what it looks like in the database and whether the transaction has been saved. Here we can see that all data match, i.e.: order numbers, payment ID, products ID and ID of the user who placed the order.



2.5.3. How I performed End-to-End Tests

End-to-End tests were performed by checking all the functionalities available so far in my application, e.g. by testing whether the application is able to register the user correctly, whether the application is able to correctly log the user in, whether I am able to make a video call using video-chat, etc.

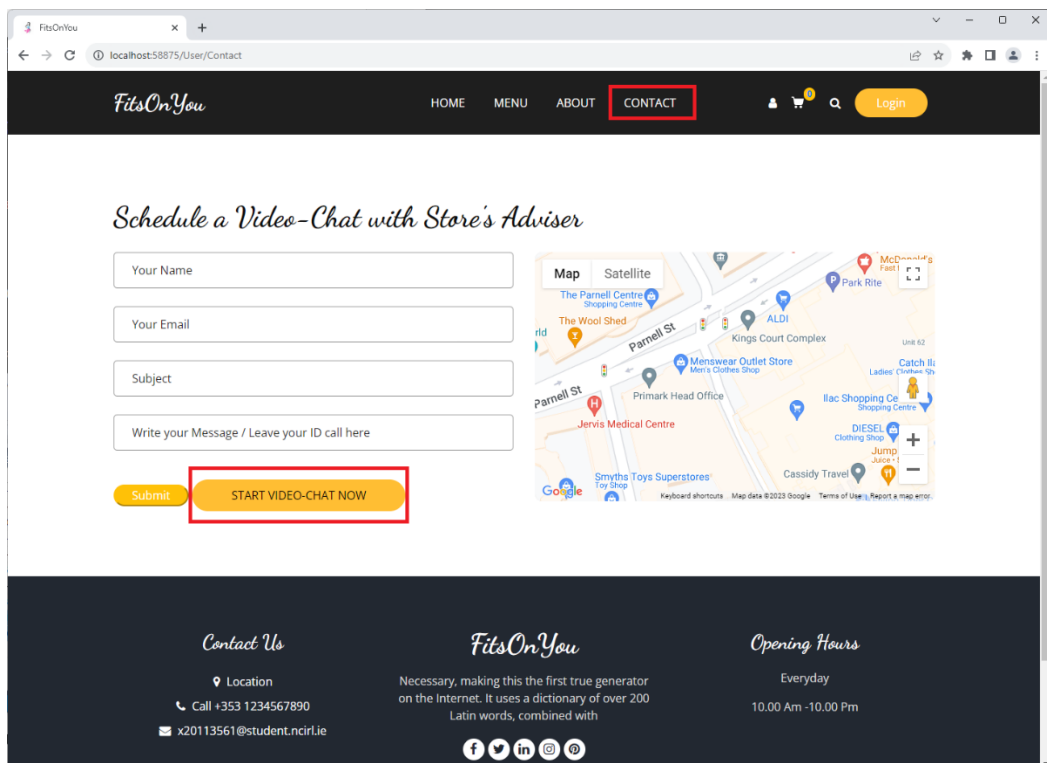
Example of End-to-End test:

Checking if making a video-chat connection works.

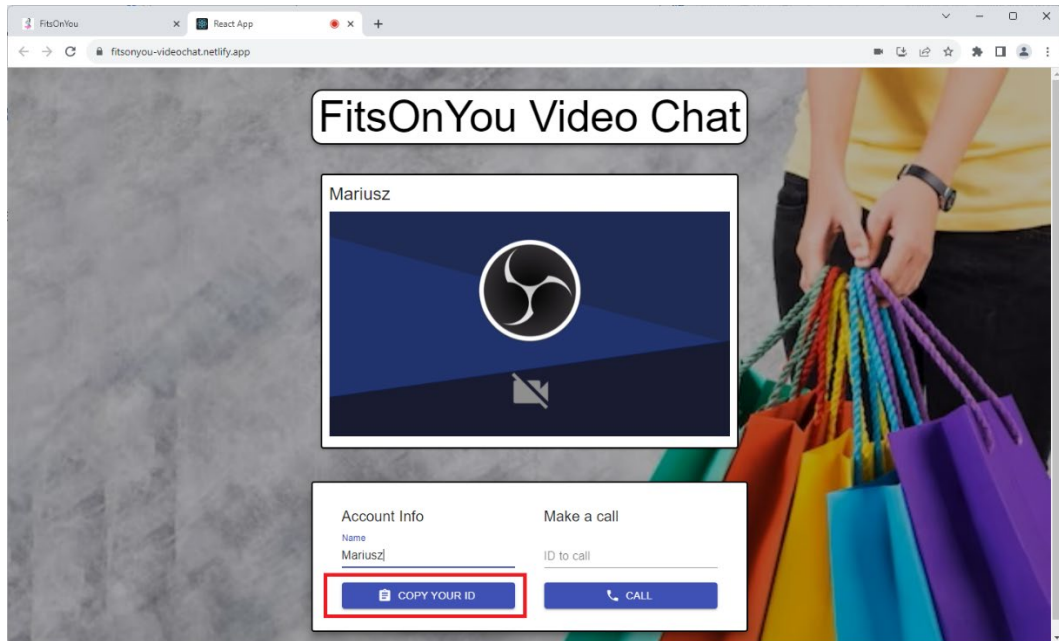
Devices used: smartphone and laptop

Testing steps:

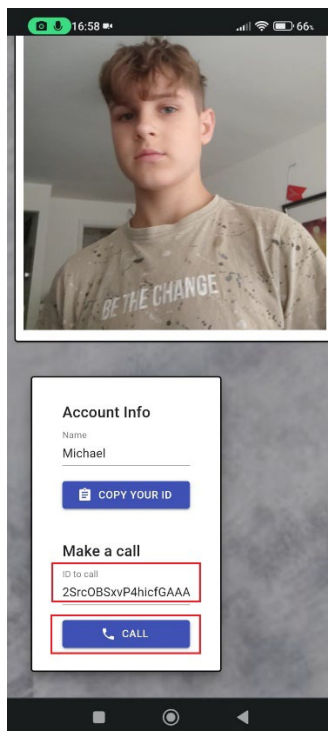
1. Click the “start video-chat now” button



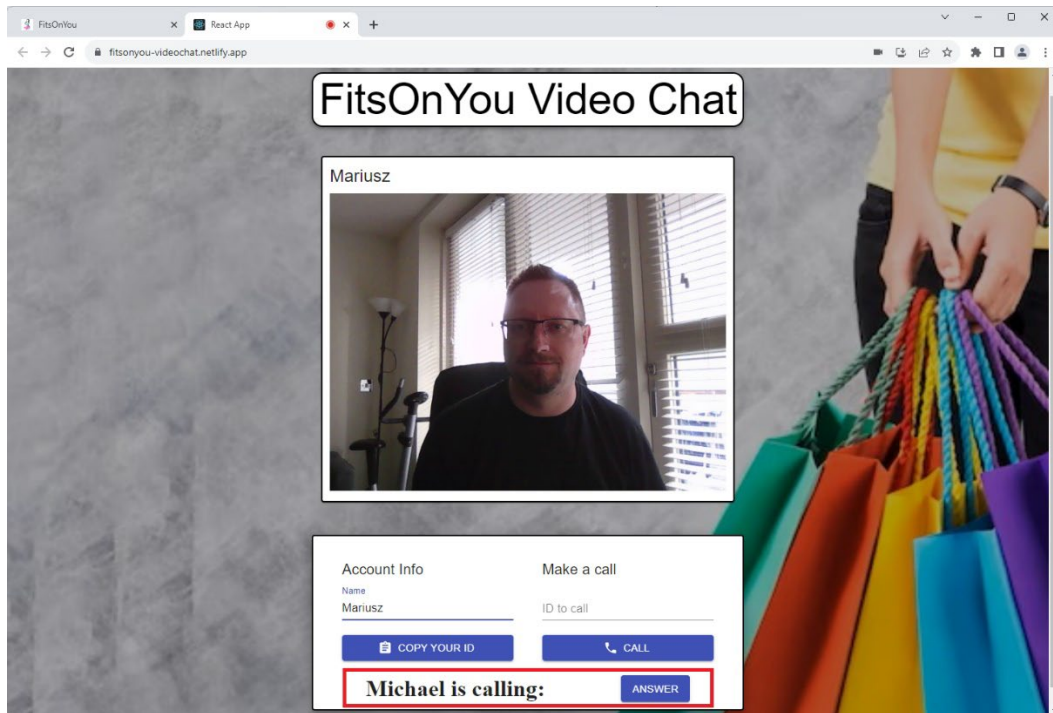
2. Generating a unique code and sending it to the person who will initiate the connection.



3. Paste the code into the "ID to call" field by the person initiating the call.
4. Press "CALL" button

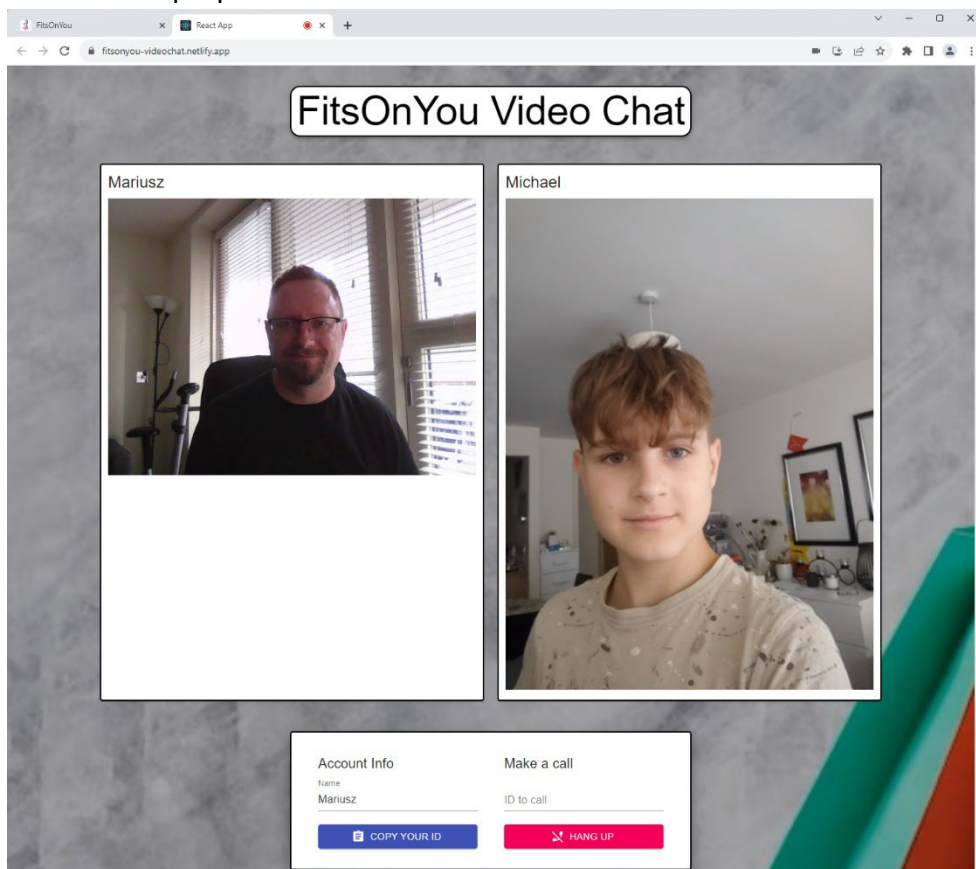


5. Press "Answer" to answer the video call

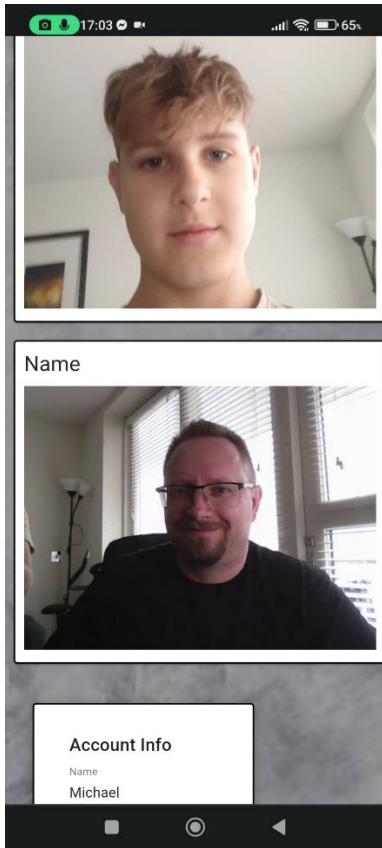


6. Establishing a connection and starting a video-chat

View from laptop:

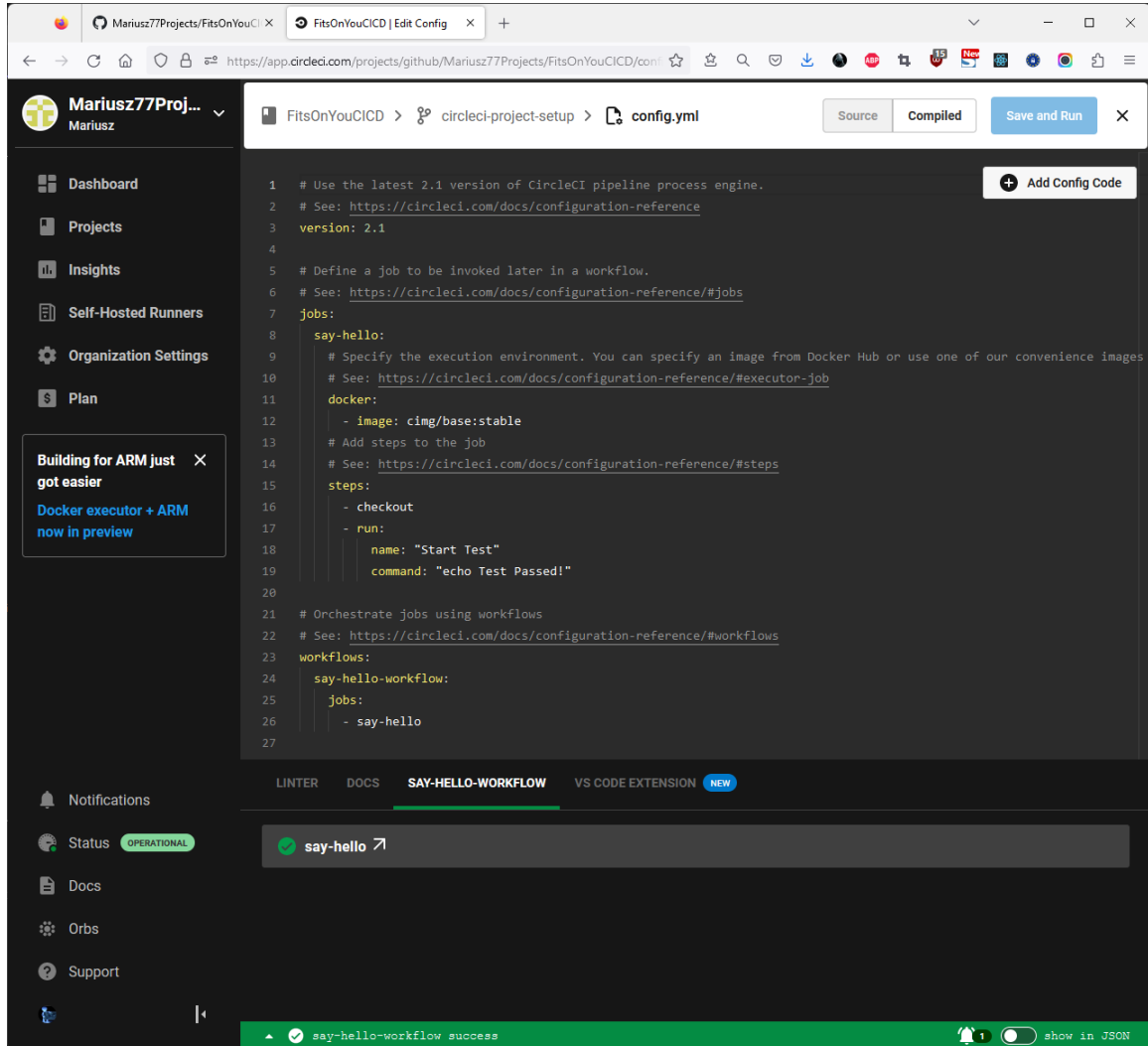


View from smartphone:



CI/DC

Below is a simple example a Pipelines CI/CD workflow used in CircleCI platform that I just used to demonstrate the feasibility of using CD/CD for my project. Of course, for build and various types of tests, more complex queries can be constructed in config.yml. Please note that this is just a basic example, and your actual pipeline might require more customization based on your project's specific needs.



The screenshot shows the CircleCI web interface for editing a configuration file. The browser address bar shows the URL: `https://app.circleci.com/projects/github/Mariusz77Projects/FitsOnYouCICD/config`. The page title is "FitsOnYouCICD > circleci-project-setup > config.yml". The main content area displays the following YAML code:

```
1 # Use the latest 2.1 version of CircleCI pipeline process engine.
2 # See: https://circleci.com/docs/configuration-reference
3 version: 2.1
4
5 # Define a job to be invoked later in a workflow.
6 # See: https://circleci.com/docs/configuration-reference/#jobs
7 jobs:
8   say-hello:
9     # Specify the execution environment. You can specify an image from Docker Hub or use one of our convenience images
10    # See: https://circleci.com/docs/configuration-reference/#executor-job
11    docker:
12      - image: cimg/base:stable
13    # Add steps to the job
14    # See: https://circleci.com/docs/configuration-reference/#steps
15    steps:
16      - checkout
17      - run:
18          name: "Start Test"
19          command: "echo Test Passed!"
20
21 # Orchestrate jobs using workflows
22 # See: https://circleci.com/docs/configuration-reference/#workflows
23 workflows:
24   say-hello-workflow:
25     jobs:
26       - say-hello
27
```

Below the code editor, there is a status bar showing "say-hello" with a green checkmark and a link icon. At the bottom of the interface, a green notification bar displays "say-hello-workflow success".

Figure 63. Workflow file (config.yml)

Below are presented all the Pipelines I used in CircleCI platform. The created project within the CircleCI platform is integrated with my project's repository on GitHub.

The screenshot shows the CircleCI interface for the 'FitsOnYouCICD' project. A maintenance alert is visible at the top. The main area displays a table of pipelines with the following data:

Pipeline	Status	Workflow	Branch / Commit	Start	Duration	Actions	
FitsOnYouCICD 5	Success	say-hello-workflow	circleci-project-setup a9874d9 Updated config.yml	2m ago	17s	Refresh, Cancel, Close	
Jobs		say-hello 3			14s		
FitsOnYouCICD 4	Failed	say-hello-workflow	circleci-project-setup 074f0fb Updated config.yml	23m ago	11s	Refresh, Cancel, Close	
FitsOnYouCIC 3	Warning	There are no workflows or build jobs in the config.					
FitsOnYouCIC 2	Error	ERROR IN CONFIG FILE: [#] required key [jobs] not found					
FitsOnYouCICD 1	Success	say-hello-workflow	circleci-project-setup 25fd82c	44m ago	13s	Refresh, Cancel, Close	

Figure 64. Example of CI/CD simple Pipelines workflow

Here are the built and testing details for this Pipeline.

The screenshot displays the CircleCI interface for a pipeline named 'say-hello'. The pipeline is in a 'Success' state, indicated by a green checkmark. The duration is 13s / 14m ago, and it was executed on a Docker / Large resource class. The branch is 'circleci-project-setup' and the commit is 'a9874d9'. The author is 'Updated config.yml'.

The pipeline consists of several steps, with the first two visible:

- Spin up environment** (10s): This step logs system information and container setup details. The logs show the build-agent version (1.0.195675-e9c9c79a), server version (20.10.18), storage driver (overlay2), backing filesystem (xfs), cgroup driver (cgroupfs), kernel version (5.15.0-1039-aws), operating system (Ubuntu 20.04.6 LTS), and architecture (x86_64). It also shows the container 'cim/base:stable' being pulled from Docker Hub.
- Preparing environment variables** (0s): This step logs the environment variables used for the build. The variables include: BASH_ENV=/tmp/.bash_env-64cab688e30f322b8be22856-0-build, CI=true, CIRCLECI=true, CIRCLE_BRANCH=circleci-project-setup, CIRCLE_BUILD_NUM=3, CIRCLE_BUILD_URL=https://circleci.com/gh/Mariusz77Projects/FitsOnYouCICD/3, CIRCLE_JOB=say-hello, CIRCLE_NODE_INDEX=0, CIRCLE_NODE_TOTAL=1, CIRCLE_PROJECT_REPONAME=FitsOnYouCICD, CIRCLE_PROJECT_USERNAME=Mariusz77Projects, CIRCLE_REPOSITORY_URL=git@github.com:Mariusz77Projects/FitsOnYouCICD.git, CIRCLE_SHA1=a9874d9ced7ce4fed278cecF64e3dc1178f10b7e, CIRCLE_SHELL_ENV=/tmp/.bash_env-64cab688e30f322b8be22856-0-build, CIRCLE_USERNAME=bregr77, CIRCLE_WORKFLOW_ID=15e22603-8f29-4e65-bc48-3abf2ac186cd, CIRCLE_WORKFLOW_JOB_ID=5fcebad1-d3f7-4d2e-acf4-1b096711ad60, CIRCLE_WORKFLOW_WORKSPACE_ID=15e22603-8f29-4e65-bc48-3abf2ac186cd, and CIRCLE_WORKING_DIRECTORY=~/project. It also shows that environment variables from project settings and/or contexts are used, with CIRCLE_OIDC_TOKEN and CIRCLE_OIDC_TOKEN_V2 being redacted.

Figure 65. Built and testing details for pipeline (1/2)

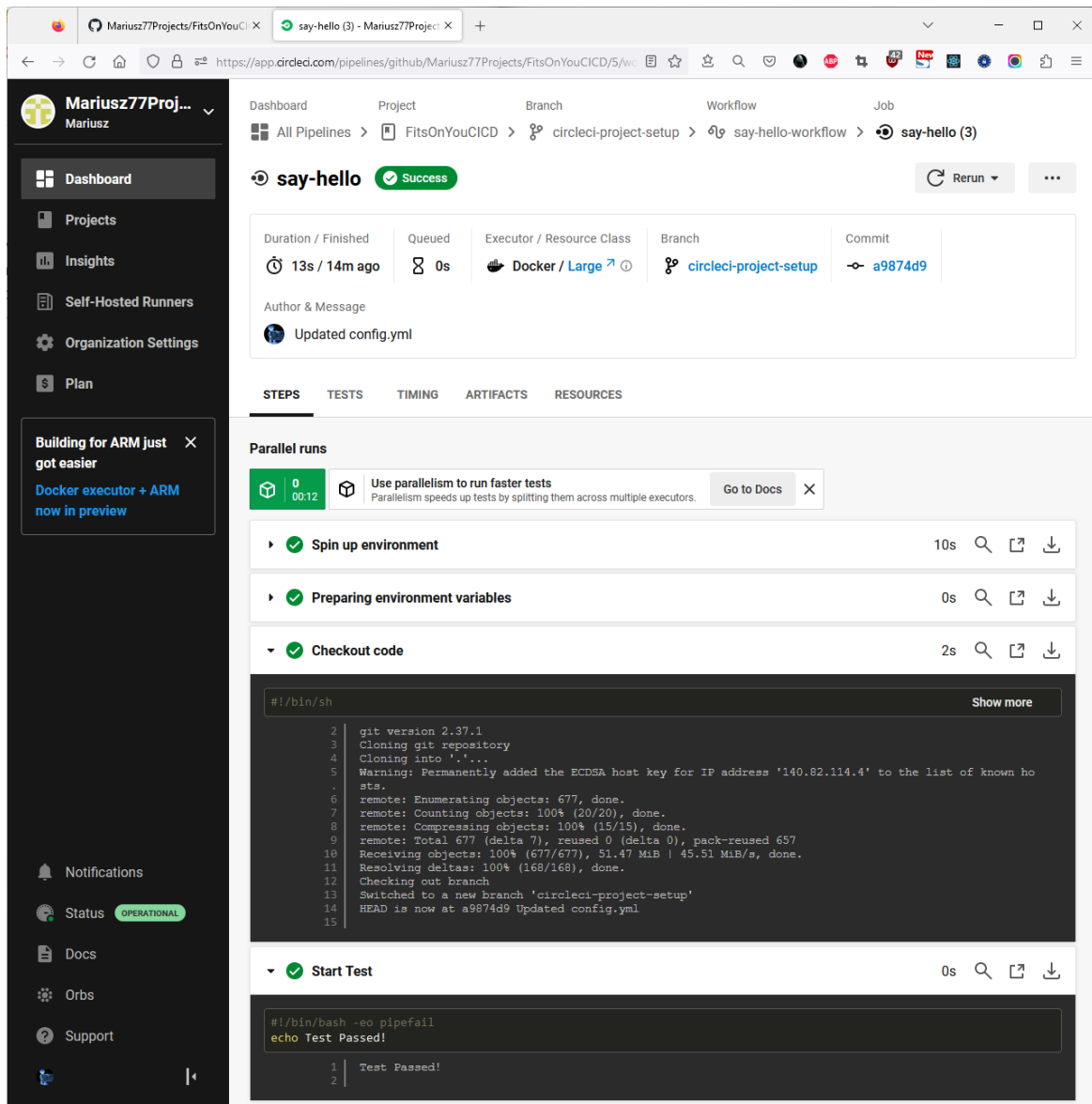


Figure 66. Built and testing details for pipeline (2/2)

2.6. Evaluation

The primary objective of this application is to enhance the online shopping experience by allowing customers to interact with support staff, stylists in real-time through video communication. The evaluation will focus on various aspects of the application, including user experience, functionality, security, and overall performance. In addition, the evaluation will provide valuable insights into the overall performance, functionality, and user experience of the E-commerce fashion store with built-in video chat. Based on the findings, recommendations for enhancements and optimizations can be made to ensure the application meets the requirements of modern online shoppers and provides a seamless video shopping experience.

User Interface and Experience:

The user interface of the e-commerce fashion store should be intuitive, visually appealing, and easy to navigate. It should allow customers to seamlessly browse products, filter items, and initiate video chats effortlessly. The interface should also be responsive, providing a consistent experience across different devices and screen sizes.

Video Chat Functionality:

The core feature of the application is the built-in video chat. The evaluation will assess the quality of video and audio transmission, latency, and reliability during video calls. It should provide clear and smooth communication, without frequent disconnections or glitches.

Security and Privacy:

As the application involves real-time communication, security and privacy are paramount. The evaluation will examine measures taken to protect user data, secure video chats, and prevent unauthorized access.

Customer Support and Assistance:

The effectiveness of customer support provided through video chat will be evaluated. This includes the availability of support staff, their knowledge, responsiveness, and problem-solving capabilities. The ability to assist customers with their purchase decisions or any technical issues during video calls will be assessed.

Performance and Scalability:

The evaluation will analyse the application's performance under various loads, including concurrent video chat sessions. The application should be able to handle multiple users engaging in video chats without significant degradation in performance. Scalability should be considered to accommodate increasing user demands.

Usability Testing with Users:

Conducting usability testing with a group of real users will be an essential part of the evaluation. Feedback from users will help identify potential pain points, usability issues, and areas for improvement.

3.0 Conclusions

The application of an e-commerce store with clothes brings both advantages and disadvantages to the table. One of the key advantages is the convenience it offers to customers. They can browse through a wide range of clothing items from the comfort of their homes, saving time and effort compared to traditional brick-and-mortar shopping, and video chat is especially helpful here. It provides professional customer service and advice on purchasing matters as well as quick problem solving. This accessibility also opens up the store to a global customer base, breaking geographical barriers and expanding the potential market. Another advantage is the ability to offer a vast selection of products. Unlike physical stores limited by shelf space, an e-commerce store can showcase a diverse inventory, providing customers with more choices to find their desired styles, sizes, and colours. This extensive selection can cater to different tastes and preferences, attracting a broader audience. Furthermore, an e-commerce store can

leverage data analytics to understand customer behaviour better. By tracking user interactions and purchase patterns, the store can personalize recommendations, promotions, and marketing campaigns, enhancing customer engagement and increasing the likelihood of repeat business.

On the other hand, e-commerce stores also face certain challenges and disadvantages. One significant drawback is the lack of a physical touchpoint. Customers cannot physically try on the clothes before making a purchase, which can lead to uncertainty and potential returns. This could result in a higher rate of product returns and added logistical costs for the store.

Strengths of the project would include the integration of high-quality product images and detailed descriptions, which can compensate for the lack of physical presence. Employing professional photography and informative content can give customers a comprehensive view of the products, simulating an in-store experience.

The project may face limitations related to technological barriers. Not all customers might have access to stable internet connections or advanced devices, potentially limiting their ability to browse the store or make purchases. Ensuring a mobile-friendly design and optimizing the website for various devices and internet speeds is crucial to mitigate this limitation.

The project of an e-commerce store with clothes presents numerous advantages such as convenience, global reach, and data-driven personalization. At the same time, it must overcome challenges like the lack of physical interaction, potential returns, and the need for robust customer service. By leveraging strengths like high-quality visuals and influencer marketing while addressing limitations through user-friendly design and inclusive accessibility, the project can strive for success in the competitive online retail market.

4.0 Further Development or Research

After minor changes to the front-end, the application can be used for online sales of other products, and the sale of clothes is only one of the examples of using this application. The project of an e-commerce store application that can be easily adapted for online sales of various products brings about several advantages and opportunities. By making minor changes to the front-end, the application gains the flexibility to cater to different product categories, allowing it to reach a broader customer base and diversify its offerings.

One of the significant advantages of this project is the potential for rapid expansion and scalability. With the capability to accommodate other product types beyond clothes, the e-commerce store can quickly pivot and tap into new markets as opportunities arise. This adaptability grants the business a competitive edge in a dynamic and ever-changing market.

The project's greatest strength lies in its versatility and potential for innovation. By being adaptable to various product types, it remains relevant in a rapidly evolving market, where consumer preferences and trends can change swiftly. This flexibility allows the store to stay ahead of competitors and seize emerging opportunities.

The project of an e-commerce store application capable of adapting to diverse product categories offers numerous advantages in terms of scalability, cost-effectiveness.

An important element of the application that he wanted to implement in his application are sales data analyses for the administrator or owner of the online store. Sales analysis is a powerful tool that enables e-commerce administrators to understand customer behaviour, identify trends, and optimize business strategies. Sales analysis provides invaluable insights into customer behaviour by analysing various metrics, such as purchase frequency, order value, and product preferences. Administrator can track which products are the most popular, which ones generate the most revenue, and which ones need improvement. Armed with this information, owner of the store can make data-driven decisions about inventory management, product recommendations, and marketing campaigns to cater to their customers' needs effectively.

In the rapidly evolving landscape of online retail, staying ahead of the competition requires innovative solutions that enhance the customer experience. To meet this, in addition to the already operating video-chat and the possibility of consulting a store assistant or purchase advisor, I would like to introduce the function of a virtual fitting room. They have emerged as a powerful tool for online clothing stores, allowing customers to try on garments virtually before making a purchase. Virtual fitting rooms address the primary concern of many online shoppers - the uncertainty of fit and appearance. One of the most frustrating aspects of purchasing clothes online is the lack of physical interaction with the product. Virtual fitting rooms bridge this gap by enabling customers to visualize themselves wearing the clothes in real-time. By simply uploading a photo or using a webcam, customers can virtually "try on" different outfits, allowing them to make well-informed decisions based on how the garments fit their body shape and size. As a result, virtual fitting rooms reduce the chances of dissatisfaction due to misfitting items, leading to lower return rates and increased customer satisfaction.

5.0 References

- [1] D. Zanzalari, "Advantages of E-Commerce," 13 09 2022. [Online]. Available: <https://www.thebalancemoney.com/advantages-of-ecommerce-1141610>.
- [2] Q. Malloy, "Benefits of E-commerce For Customers and Businesses," 22 01 2019. [Online]. Available: <https://www.cloudtalk.io/blog/benefits-of-e-commerce-for-customers-and-businesses/>.
- [3] Microsoft, "ASP.NET," Microsoft, [Online]. Available: <https://dotnet.microsoft.com/en-us/apps/aspnet>.
- [4] Microsoft, "Visual Studio Community," [Online]. Available: <https://visualstudio.microsoft.com/vs/community/>.
- [5] M. Rouse, "What Does ASP.NET Mean?," 28 October 2012. [Online]. Available: <https://www.techopedia.com/definition/3213/asp-net>.
- [6] Microsoft, "ASP Overview," Microsoft, 16 06 2017. [Online]. Available: [https://learn.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms524929\(v=vs.90\)](https://learn.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms524929(v=vs.90)).
- [7] w3schools, "ASP and ASP.NET Tutorials," [Online]. Available: <https://www.w3schools.com/asp/>.
- [8] Alibaba Cloud Bao, "What is future scope of iis web server," 2023. [Online]. Available: <https://www.alibabacloud.com/tech-news/web-server/giqt1yoqzk-what-is-future-scope-of-iis-web-server>.
- [9] S. J. B. Linda Rosencrance, "Internet Information Services (IIS)," 2019. [Online]. Available: <https://www.techtarget.com/searchwindowsserver/definition/IIS>.
- [10] Microsoft, "Designing ASP Applications," 2017. [Online]. Available: [https://learn.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms525124\(v=vs.90\)](https://learn.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms525124(v=vs.90)).
- [11] Microsoft, "Optimizing the Performance of Database Access in IIS," 2017. [Online]. Available: [https://learn.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms525484\(v=vs.90\)](https://learn.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms525484(v=vs.90)).
- [12] E. Seyer, "Core Differences Between IIS and the ASP.NET Development Server (C#)," 2022. [Online]. Available: <https://copyprogramming.com/howto/core-differences-between-iis-and-the-asp-net-development-server-c>.
- [13] Star Knowledge, "How to Select the Right Tech Stack for Web Application Development," 2022. [Online]. Available: <https://star-knowledge.com/blog/best-tech-stack-for-web-app-development/>.
- [14] A. H. Jessica Sirkin, "Microsoft SQL Server Management Studio (SSMS)," 2 2017. [Online]. Available: <https://www.techtarget.com/searchdatamanagement/definition/Microsoft-SQL-Server-Management-Studio-SSMS>.

- [15] C. S. Adam Hughes, "Microsoft SQL Server," 06 2019. [Online]. Available: <https://www.techtarget.com/searchdatamanagement/definition/SQL-Server>.
- [16] Microsoft, "System stored procedures (Transact-SQL)," Microsoft, 26 05 2023. [Online]. Available: <https://learn.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/system-stored-procedures-transact-sql?view=sql-server-ver16>.
- [17] TutorialTeacher, "SQL Server - Stored Procedures," TutorialTeacher, 2023. [Online]. Available: <https://www.tutorialsteacher.com/sqlserver/stored-procedures>.
- [18] A. Olanrewaju, "SQL Server - CRUD operation using stored procedure," 26 10 2020. [Online]. Available: <https://hashnode.com/post/sql-server-crud-operation-using-stored-procedure-for-beginners-part-1-ckgq94lz209lgncs1c7nwcu8j>.
- [19] S. Cletus, "Implementing Video Conferencing with React, Node.js, and WebRTC," 20 08 2021. [Online]. Available: <https://www.section.io/engineering-education/video-conferencing-app-with-react-node/>.
- [20] M. Wargowski, "WebRTC Node.js tutorial: Development of a real-time video chat app," 07 07 2023. [Online]. Available: <https://tsh.io/blog/how-to-write-video-chat-app-using-webrtc-and-nodejs/>.
- [21] P. S. A. f. Beginners, "Myron Dsilva," 2022, 27 06. [Online]. Available: <https://www.analyticsvidhya.com/blog/2022/06/python-stock-analysis-for-beginners/>.
- [22] I. Shah, "Stock Market Data: Obtaining Data, Visualization & Analysis in Python," 20 07 2021. [Online]. Available: <https://blog.quantinsti.com/stock-market-data-analysis-python/>.
- [23] S. J. Bigelow, "What goes into a user story vs. use case for Agile development?," 25 Oct 2019. [Online]. Available: <https://www.techtarget.com/searchsoftwarequality/answer/What-is-the-difference-between-a-user-story-and-use-case-in-software-testing>.
- [24] Y. Francino, "What is a user story?," November 2022. [Online]. Available: <https://www.techtarget.com/searchsoftwarequality/definition/user-story>.
- [25] v. paradigm, "User Story vs Use Case," visual paradigm, [Online]. Available: <https://www.visual-paradigm.com/guide/agile-software-development/user-story-vs-use-case/>.
- [26] S. Blake, "Use Cases vs. User Stories: How They Differ and When to Use Them," 08 Jun 2021. [Online]. Available: <https://www.easyagile.com/blog/use-cases-vs-user-stories/>.
- [27] S. T, "Use Case vs User Story: Everything You Need to Know," 13 Apr 2023. [Online]. Available: <https://www.knowledgehut.com/blog/agile/use-cases-how-are-they-different-from-user-stories-how-to-create-them>.

6.0 Appendices

6.1. Project Proposal

6.1.1. Objectives

Thanks to my application “Fits on you”, you will be able to make online purchases of cloths and accessories in a quick and easy way. The application will have a clear interface and navigating the website and the application will be intuitive.

The application will provide a large selection of goods, where compared to traditional brick-and-mortar stores, the choice is limited due to the rising prices of storage. The advantage of the online store from the customer's point of view is the fact that in this type of store you can see a greater tendency to discount stores and price promotion, which contributes to attracting customers to such a store.

I want to create an application, the skeleton of which would be multitasking, suitable for all kinds of applications as part of creating a product database, sales, status of products shipped to the customer, whether the goods have been shipped, are in transit or whether they have been picked up by the customer. I want to focus on making the basis and framework of the project allow the application to be used in many aspects of sales or running services that can be implemented online.

Another innovation will be the possibility of enabling video-consulting from the store via my application. The customer will be able to contact the store department, where the stylist or other specialist will be able to advise on the selection of the right clothes via a video conference built into my application.

The video conference included in the sales application is not only a convenient addition for the customer, but also increases the prestige of the store, which in such a situation is perceived as more professional.

6.1.2. Background

My project aims to create an application that will allow you to make purchases in an easy and pleasant way, which will make it a good idea to attract more customers, which will make the application a sought-after product for people and companies who want to open new businesses or increase the sale of your products on the Internet. This is to attract the most demanding and undecided customers to the store and encourage them to buy.

6.1.3. State of the Art

There are a lot of applications on the market for online shopping, but the customer is only reliant on what he sees in the form of a static image on the website, where in such a situation it is often difficult to decide whether a given product is appropriate and whether the customer will be satisfied with the purchase. One of such applications and services is, for example, Zalando, where you can buy clothes, footwear, accessories, household items, etc. The choice of goods is very large, you can hunt down products of well-known brands at lower

prices. The application has a very user-friendly interface as well as an intuitive menu. However, it is an application that does not have a built-in video chat function and contact with a stylist, which would help in choosing a product and making the appropriate purchase by the customer. Often, customers need the opinion of specialists who would be able to professionally advise on the selection of the product that best suits the customer.

6.1.4. Technical Approach

The application will consist of two main modules (user module and admin module), in which we will be able to distinguish part of the application intended for the customer and the other part for the owner of such a store.

Priorities for the operation of the part of the application visible to the customer:

- ease and intuitiveness of application navigation
- quick finding of the product you are looking for thanks to the search engine and dividing products into categories
- ease of making transactions
- possibility of printing the invoice in pdf format of all purchases made, saved in the history of purchases in the customer's profile
- tracking your shipment after making a purchase
- possibility to post feedback

Features and elements of the application included in the module intended for the store owner:

- Easily create, add, and edit product categories
- Managing and updating the quantity of available products
- Easy access to create product description and details
- Custom display, depending on the needs, of the database of products, customers and transactions
- Access to the database containing customer feedback

6.1.5. Technical Details

Choosing the right technology is the key to the success of the entire project. My choice of technology is dictated by the client's needs and the specificity of the business, which in my case is online sales. I am going to create my project based on ASP.NET, which is a set of technologies based on a framework designed by Microsoft. For this purpose, I will be using the free development environment Visual Studio Community Edition.

ASP.NET (Active Server Pages.NET) is an environment that allows you to easily create dynamic Internet applications, home pages or online stores. It connects various kinds of elements of the WWW. The technology was developed by Microsoft.

ASP.NET is the latest version of the well-known ASP technology. Its new version is very different from the classic one. The ASP.NET programming model is transformed and improved and includes many new tools that make it much easier to design and create dynamic web pages. One of the benefits of this technology is that it provides powerful tag controls. They are very similar to HTML tags. ASP.NET provides many new possibilities and extensions, it allows you to display randomly selected ads or a calendar. Thanks to the control elements, the programmer can generate complex HTML code corresponding to the applicable standards. The work required for this activity is minimal. The ASP.NET technology allows you to use the information contained in databases and allows you to adjust the website depending on the preferences of the users who visit it.

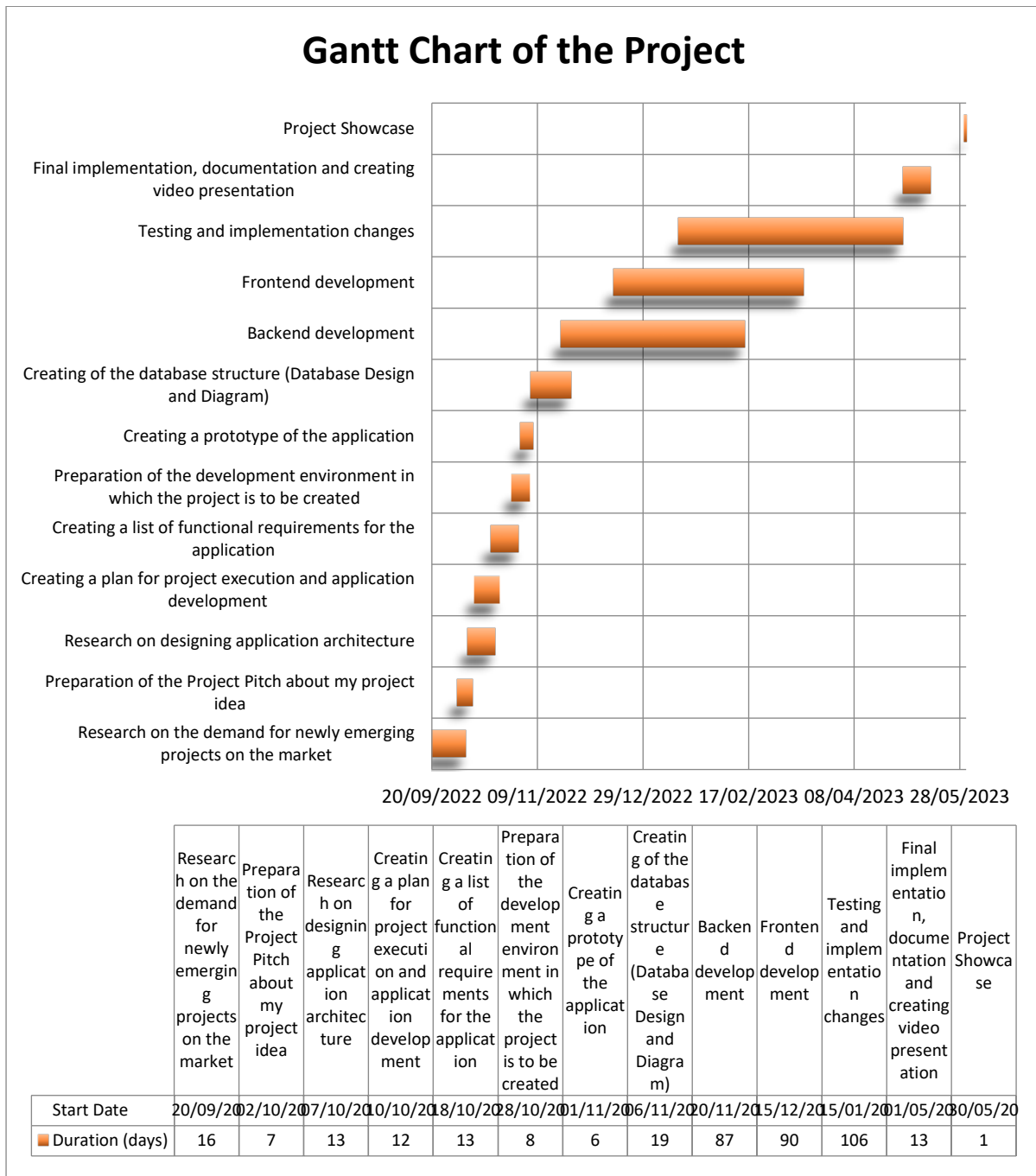
Currently, ASP.NET offers three frameworks for building web applications: Web Forms, ASP.NET MVC (Model-View-Controller) and ASP.NET Web Pages. All three platforms are stable and perfect for web application development. For my purposes, I believe that the Web Forms version will be optimal. With ASP.NET Web Forms, you can create dynamic websites using the familiar drag-and-drop event model.

The database will be created using SQL Server Management Studio and get stored on SQL Server.

6.1.6. Special Resources Required

No additional external special sources will be used in the project. All data entered will not be based on existing real data. These data will only be testing data entered by me for testing purposes. Here the idea may change, depending on whether, for testing purposes, at a later stage of building the application, I would like to add an API of a real working store.

6.1.7. Project Plan



Below are the starting dates for each phase of the project. Most of the dates given here are only assumptions and may constitute a reference to the assumed goals related to the time of the project's development. I reserve that they may be changed.

20 Sep 2022 - Research on the demand for newly emerging projects on the market

Research and development of the idea is the basic element influencing the overall project.

02 Oct 2022 - Preparation of the Project Pitch about my project idea

Whether my idea is sold, and I can start the project depends on how I present my idea.

07 Oct 2022 - Research on designing application architecture

Here I must go over how projects like the one I am going to work on are carried out in terms of the different technologies currently used and the technologies that could be used.

10 Oct 2022 - Creating a plan for project execution and application development

Creating a web application is included in a well-planned, comprehensive process whose main goal is to deliver a product that meets the intended goals.

18 Oct 2022 - Creating a list of functional requirements for the application

The functional requirements of the application describe the behaviour of the system under certain conditions and define the features and functions of the product. These requirements include technical details, data processing and authentication

28 Oct 2022 - Preparation of the development environment in which the project is to be created

Preparation of the development environment will include the installation of several tools that I will use during the project development.

01 Nov 2022 - Creating a prototype of the application

Bearing in mind the conclusions from the earlier stages, I start the application development process, in which I create the skeleton and the early version of the application, i.e., the prototype.

06 Nov 2022 - Creating of the database structure (Database Design and Diagram)

Creating the database skeleton, dependencies, and relationships of all database elements.

20 Nov 2022 - Backend development

15 Dec 2022 - Frontend development

15 Jan 2023 – Testing and implementation changes

Testing the application will take place simultaneously with the work on the code from the beginning of the code development and at every stage of the application stage. Of course, after completing the work on the application, it is necessary to comprehensively test the whole thing.

01 May 2023 - Final implementation, documentation and creating video presentation

This is the time for the last fixes and implementation of changes in the application. However, I assume that this will only be polishing the application in the form of minor tweaks, not affecting the overall performance of the application and the design.

30 May 2023 - Project Showcase

The final step of the project

6.1.8. Testing

Running tests are extremely important. The first and most obvious reason is finding hidden bugs. Although at first glance it may seem that the application is fine, often errors appear over time and, unfortunately, are found by customers. And this can have very unpleasant consequences, both financial and image related. I am going to conduct tests with the participation of a group of people who will share their observations with me and will report all comments regarding the application. I intend to carry out the tests at various stages of the project, in order to detect and eliminate errors as soon as possible or introduce the necessary corrections.

The scope of the tests will include:

- **Testing the homepage**
One of the most important parts of any website is what appears first to users. The home page must be well structured for the user's eye, but not only. In addition to its appearance, functionality is also important. In order to improve it, the tester must check each of its elements to see if it works as, it should.
- **Ecommerce Optimization Checking**
Another factor that I need to check is the functioning of all elements of the website that perform important functions, e.g., sell products. I need to check that all the attributes of my application are compatible with each other. It is about synergy between all elements of the website: subpage of the offer, products, category sections, etc. Lack of well-matched subpages will result in a decrease in product sales.
- **Filter test on the site**
Filters are a feature that customers use very often. It is important that in the e-commerce application all filters, product and category search processes work smoothly. Problems with such elements of your store may arise after introducing new content to the store, new products, categories, or too many reviews of certain products.
- **Payment gateway testing**
When checking out my application, keep in mind the area that finalizes the customer's purchases. Testing payment gateways is extremely important due to their access to sensitive data; both the store and the customer. The tester will have to operate such gates, and I would not like all payment segments to be checked; from price details to invoicing.
- **Security testing**
Security testing is extremely important for any online store. As in many cases, my application will also store data about consumers; their bank accounts, contact details. Hacker attacks are appearing more and more frequently on the Internet, especially on smaller and less secure stores. Therefore, I need to make sure that this part of my

project has the appropriate security features. In this case, it might be a good idea to make pentests.

6.2. Ethics Approval Application (only if required)

6.3. Reflective Journals

6.4. Invention Disclosure Form (Remove if not completed)

6.5. Other materials used

6.6. Project Plan

+ Add Expand all Collapse all Zoom in Zoom out Zoom to fit						
ID ↑	Name	Start Date	End Date	Duration	Progress %	
1	▼ Plan	Sep 20, 2022	Oct 22, 2022	33 days	100	
2	Research on the demand for newly emerging projects on the market	Sep 20, 2022	Oct 05, 2022	16 days	100	
3	Preparation of the Project Pitch	Oct 02, 2022	Oct 09, 2022	8 days	100	
4	Research on designing application architecture	Oct 07, 2022	Oct 20, 2022	14 days	100	
5	Creating a plan for project execution and application development	Oct 10, 2022	Oct 22, 2022	13 days	100	
6	▼ Technology	Oct 12, 2022	Oct 20, 2022	9 days	100	
7	Find Examples	Oct 12, 2022	Oct 17, 2022	6 days	100	
8	Test Examples	Oct 15, 2022	Oct 20, 2022	6 days	100	
9	▼ Start Project	Oct 18, 2022	Apr 05, 2023	170 days	83	
10	Creating a list of functional requirements for the application	Oct 18, 2022	Nov 27, 2022	41 days	100	
11	Preparation of the development environment	Oct 28, 2022	Nov 09, 2022	13 days	100	
12	Creating a prototype of the application	Nov 01, 2022	Nov 19, 2022	19 days	100	
13	Creation of the database structure (Database Design and Diagram)	Nov 06, 2022	Dec 24, 2022	49 days	100	
14	Backend development	Nov 20, 2022	Jan 28, 2023	70 days	66	
15	Create UI Design	Dec 15, 2022	Mar 29, 2023	105 days	66	
16	Testing and implementation changes	Feb 10, 2023	Apr 05, 2023	55 days	100	
17	Mid Point Presentation	May 14, 2023	May 14, 2023	1 day	0	

Figure 67. Planned tasks to be carried out as part of the project development.

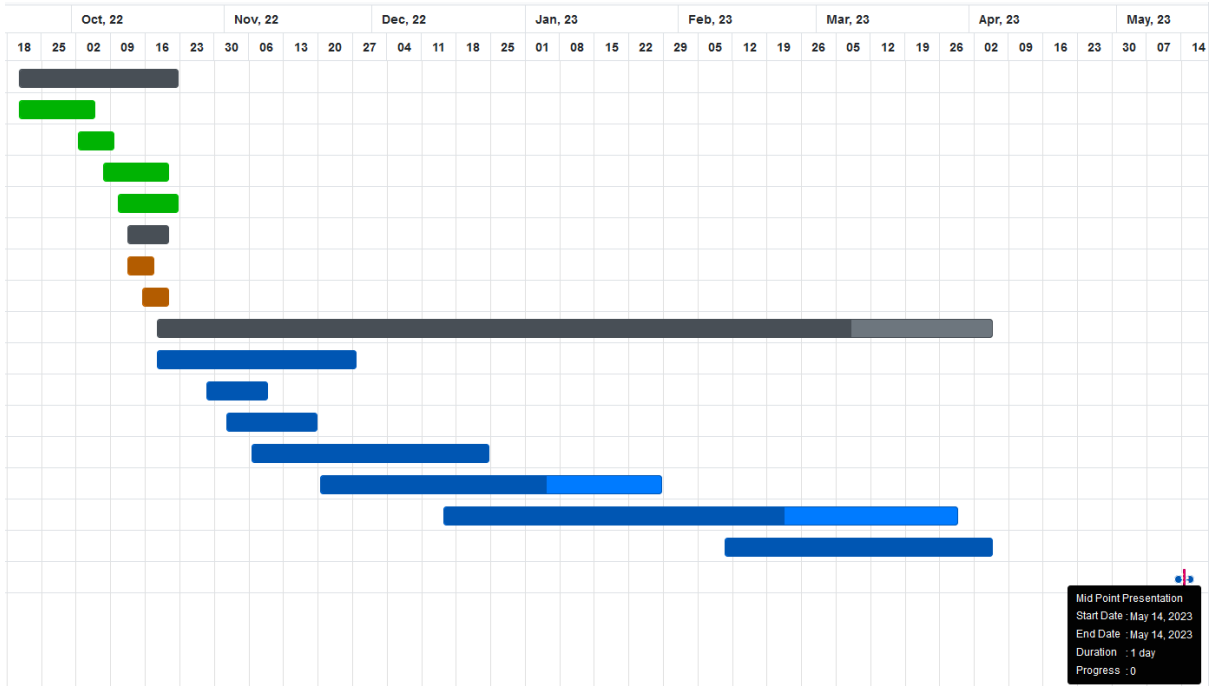


Figure 68. Tasks and milestones visualized on a timeline.

6.7. Backlog

I used monday.com portal to plan the details of tasks and stages of developing my application. It helped me stay organized and track progress throughout the project.

▼ Sprint 1 * Customers want to plot stock data to get insights into how stock will perform in the future.

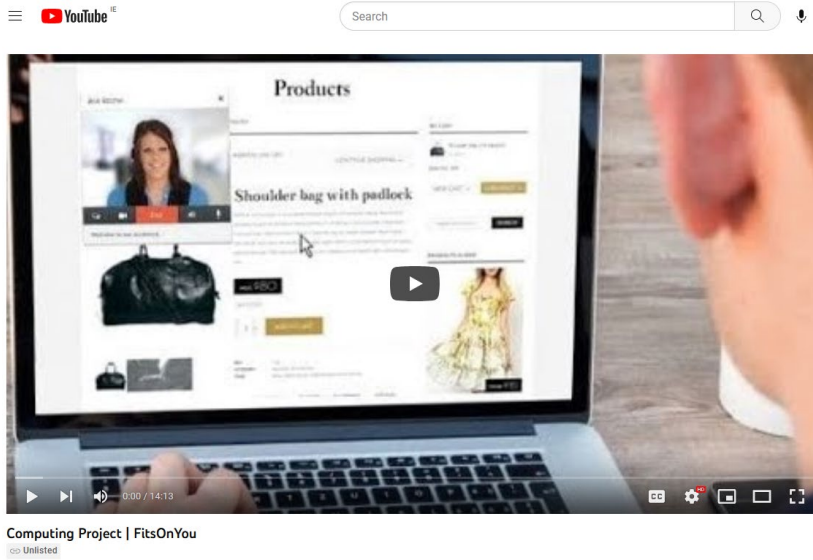
<input type="checkbox"/>	Item	Person	Status	Date	Priority
<input type="checkbox"/>	1.1 Download some stock data somewhere for us to template a database (find out schemas). (1-5...	MK	In progress	Nov 14, 2022	High
<input type="checkbox"/>	1.2 Research column-oriented databases for possible inclusion in our system (2-3 days) (2 points)	MK	Done	Nov 17, 2022	Medium
<input type="checkbox"/>	1.3 Create a database that hosts the data from 1.1 (4-5 days)	MK	Done	Nov 22, 2022	High
<input type="checkbox"/>	+ Add Item				

▼ Sprint 2 * Frontend and DB


<input type="checkbox"/>	Item	Person	Status	Date	Priority
<input type="checkbox"/>	2.1 Connecting DB to frontend	MK	Done	Feb 22	High
<input type="checkbox"/>	2.2 Populate DB	MK	Done	Feb 23	Medium
<input type="checkbox"/>	2.3 Test the DB	MK	In progress	Feb 23	Medium
<input type="checkbox"/>	2.4 Convert DB to Column	MK	In progress	Feb 24	High
<input type="checkbox"/>	2.5 Test DB in Column ver	MK	In progress	Feb 25	Medium
<input type="checkbox"/>	+ Add Item				

Figure 69. Backlog representing parts of tasks to do

6.8. Link to video presentation for Mid-Point of the project



<https://youtu.be/3IkzIDu2gbk>



Signature

Dublin, 13/05/2023

6.9. Monthly reports

Supervision & Reflection | Report No. 1

Student Name	Mariusz Kucharski
Student Number	x20113561
Course	Bachelor of Science in Computing
Supervisor	William Clifford

Month: October

What?

Reflect on what has happened in your project this month?

This month I mainly focused on the project idea. This is extremely important because the success of the entire project depends on it. The challenge was to get to know the needs of the potential client and his expectations. The idea was to create a product that would be useful and meet these expectations. After collecting feedback and analysing it, I came up with an idea that may seem innovative compared to similar and existing applications. Based on my observations, it can be concluded that the market lacks an application with an online store in which the customer could get fully professional help in choosing the product that best suits his needs. Customers like to be sure that the product they choose is the right one, or they often require help in advising the right product by an expert, who is not every customer. That is why I decided to add a video-chat function to my application, which allows to communicate with a stylist or other person professionally involved in advising on the best choice of a product.

I created the so-called Project Proposal, in which I included basic information about the project idea and the basics of my application. However, due to the initial stage of the project, I could not go into details of the project.

Later this month, 28th Oct, I managed to get in touch with my supervisor William Clifford. We discussed topics related to my project. We managed to reach a consensus consisting in the implementation of the idea related to the introduction of an innovative use of video-chat between the client and the specialist. However, due to the complexity of my previous initial idea, for the time being we have suspended its implementation consisting in building the application in such a way that the application skeleton, after minor changes in the code, can be used in other areas of online store operations.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

The biggest challenge now, I can say, is to keep gathering information about the ASP.NET Web Forms technology with which I want to develop my application. I have participated in many webinars in which training courses were conducted on creating applications using ASP.NET technology using 3 different versions of this environment: MVC, Core and Web Forms. To create my project, I decided to use the Web Forms version, as in my opinion this technology best suits the requirements of creating an application for an efficient online store. I am aware that this is only the beginning and there is a lot ahead of me, and time is passing inevitably. Therefore, I am already preparing the development environment needed to create my application.

Now What?

What can you do to address outstanding challenges?

The entire project is a unique challenge as it is individual work and teamwork cannot be relied upon. In my third year of NCI, during my 9 months of my student internship with a real and reputable .ie domain hosting company, I have not encountered a situation where any employee

was on their own to work on any project. All projects were teamwork, which I believe is the predominant nature of work in any company, and where preparing a student in this mode should take priority over the type of individual work, less common in the IT industry in the real world. Therefore, in conclusion, I can say that the challenge is to allocate time independently and create a plan for creating and developing the entire project.

Coming back to the topic, regarding my project, I agreed with my promoter that in order to start working on the project properly, in its initial phase I need to create use case diagrams and a product backlog.

Student Signature



Supervision & Reflection | Report No. 2

Student Name	Mariusz Kucharski
Student Number	x20113561
Course	Bachelor of Science in Computing
Supervisor	William Clifford

Month: October

What?

Reflect on what has happened in your project this month?

This month I focused on rethinking the innovation of my project. For the purpose of illustrating my idea and presenting it to my supervisor, I created a use case diagram that briefly shows how my project works. After discussing with the supervisor, we agreed that it would be a good idea to introduce some elements to the project that would provide more innovation. We also agreed that my application will allow users to work with stock information and provide interface to communication between customer and experts in stock. The idea seems innovative due to facilitating the planning of purchases by researching the market based on stock exchange data. I want to provide in additional future for users, like a multi dimension kind of data for the users (various stock to different variables). The client will be able to prepare various types of lists and comparisons, based on the presented in the form of graphs and plot stock data to get insights into how stock will perform in the future.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

I can say that finding an idea that can bring innovation to my project is a success, but it will also be a challenge to analyse many aspects, such as implementing the idea into the project.

The Sprint Backlog, i.e., a list of the upcoming tasks related to the project, was created in the Monday.com application, where the subsequent stages of tasks, their descriptions and proposed dates of task completion can be placed. My priority task for this moment is to download some stock data to template a database and find out schemas. The next task is research on column-oriented databases for possible inclusion in our system. These jobs are needed to complete the third task, i.e., create a database that hosts the data from the previously downloaded stock data.

Now What?

What can you do to address outstanding challenges?

A new feature regarding the introduction of new functionalities for the user requires exploring many aspects of knowledge about multi-dimensional stock data. I'm also facing the problem of parsing a less popular column-oriented database that just happens to be more suited to my project. I think it will be a challenge for me that I hope to meet.

Student Signature



Supervision & Reflection | Report No. 3

Student Name	Mariusz Kucharski
Student Number	x20113561
Course	Bachelor of Science in Computing
Supervisor	William Clifford

Month: March

What?

Reflect on what has happened in your project this month?

In this report, I would like to include the progress made in the development of my project. The application that I am creating in ASP.NET environment on has been divided into two sectors, the admin panel, and the user part.

Administrative part:

The following milestones were achieved:

- I have completed an admin desktop UI project which includes a friendly admin interface.
- I integrated the admin dashboard with the database, allowing the admin to manage both the product categories and the products themselves. I added the function of creating, updating, and deleting categories and products from the admin dashboard.

User part:

Also, here I made significant progress in the user part of the application reaching the following milestones:

- I've completed a UI design for an app that includes a modern interface for seamless user experience.
- I implemented a system for searching and filtering categories and products.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

The biggest challenge I faced was integrating the admin desktop with the database. The database is created in Microsoft SQL Server 2022 as a relational database management system (RDBMS) developed by Microsoft. To create a database in Microsoft SQL Server 2022, I had to properly configure SQL Server Management Studio (SSMS), which is the primary interface for working with SQL Server databases and then connect to the SQL Server instance where I wanted to create the database. For the full operation of the database with the administrator's desktop, it was necessary to create the appropriate queries for Category_Crud and Products_Crud in the System Stored Procedures tab in the Programmability section.

One of the main challenges in user part I faced was implementing the search and filter functionality. It took a lot of work to make sure the filters work properly, and the search results are accurate.

Now What?

What can you do to address outstanding challenges?

Next steps:

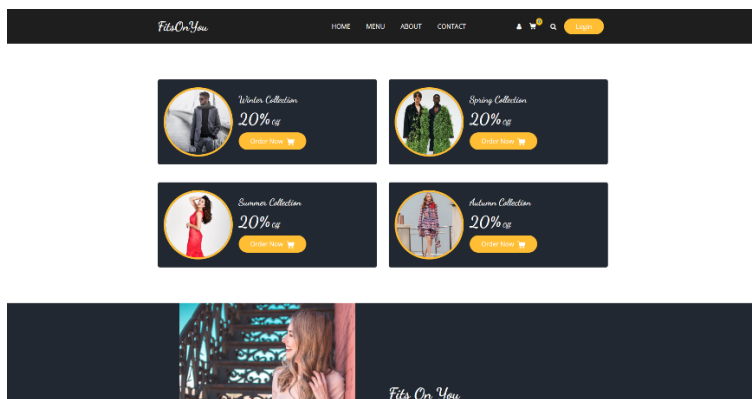
Moving forward, I plan to focus on adding more features to the admin dashboard. My goal is to add a user management system, including creating new accounts, editing existing ones, resetting passwords, and managing user roles, and implementing a selling report as an analysis of the sales plan.

There is still a lot of work to be done at this stage of project development. One of them is to complete the database connection with all elements in the Administrator panel, such as the part related to users. After completing this task, I plan to populate the DB with more entries of records to perform tests related to the performance of the database and check how the application system works with this kind of database. The next step will be to convert the database to a column-based version of the database and run the performance tests again. Running the tests will be an extremely important step in choosing the most appropriate version of the database for the type of application I am developing.

Student Signature



6.10. Link to the FINAL video presentation



<https://youtu.be/kk5kHgoXeC8>

A handwritten signature in black ink, appearing to read 'M. Anderson', written over a horizontal line.

Signature

Dublin, 07/08/2023