National College of Ireland

Software Project

Software Development

2020/2021

Krunoslav Bubanj

X18110274

x18110274@student.ncirl.ie

TeaCat System

Technical Report

Contents

Executive Summary

TeaCat System aims to provide a solution for any company that seeks to unify the standard client relation management system and a ticketing system. The idea is to combine both of them into one single entity where most of the repetitive daily time-consuming tasks will be automated.

Not only that, but also to allow its users some creative freedoms when it comes to setting up their dashboards, setting up various departments and setting up various user permissions, hence dictating who can see what within the system. This type of feature is especially important these days as the GDPR and DPA laws are mandating absolute compliance for each company dealing with the data from their customer base.

1.0 Introduction

1.1. Background

This project idea came to existence from known drawbacks of the current similar solutions. On my place on work, often enough, my work colleagues would complain how inefficient the current ticketing and customer relation management software are as they are not meeting all of our business requirements. With the expansion of the team, more and more drawbacks became apparent and there was no tailored solution that would address these issues. From this experience, I have decided to attempt and build a software that would address the most common problems our current systems have and implement it into one single solution.

1.2. Aims

The aim of the project is to create a solution for the issues arising from the drawbacks of the current systems used for the customer relation management and ticketing systems. If executed properly, the TeaCat System should allow the end user to create customisable records for the customers and internal staff, create and actions tickets that would come both internally and externally as well as automate a lot of manual tasks that usually take much of the time for the staff that could otherwise be spent more productively elsewhere.

1.3. Technology

In order to complete this project, various technological solutions will be used. Most of the code will be written in C# language as a part of ASP.NET MVC 5 while using the Entity framework. The database will be created using MSSQL and SQL for querying. The front end will be built using the HTML language and Bootstrap for the enhanced user experience, with a generous help of the Entity framework. The code will be written and tested using the Visual Studio 2019. More comprehensible details are outlined below.

IDE

The entire project so far was developed in the Visual Studio 2019 as it contains a local database where the code and functionality of the project can be tested. Visual Studio is also quite handy as it allows for scaffolding, thus saving time on some of the trivial and time-consuming tasks that would otherwise have to be done manually. Aside from aspects mentioned, Visual Studio also has a pretty useful debugger built-in allowing for the easy testing of the code.

Back End

The back end of the project is built using the C# language as a part of ASP.NET MVC 5 framework, as well as Entity Framework. These frameworks allow us to implement the separation of concerns as well as a local database. The database is MSSQL and the language used for queries is SQL.

Front End

The front end of the project is built using a combination of scaffolding features by Visual Studio Entity frameworks, Razor markup language, HTML, CSS, JQuery, JavaScript and Twitter Bootstrap.

Table of the stack used in this project

IDE	Visual Studio 2019
Programming Languages	C#
	JavaScript
	HTML
	CSS
	SQL
	Razor
Frameworks	ASP.NET MVC
	Entity Framework
Database	MSSQL
Libraries	Twitter Bootstrap
	JQuery

1.4. Structure

Introduction

Within this section, the reader will find the information about the background of the project explaining where did this idea came from. The reader will also be able to see what are the aims of this project once it is finished as well as who is it intended for.

After that, the reader will be able to see a brief overview of all technologies used to complete this project including IDE, front end languages and frameworks as well as the back end languages and frameworks.

System

The system section will probably contain the most of information, starting from the functional requirements, use case diagrams for the entire system built so far as well as the individual use case diagrams for each operation the system can perform. Each of the operations in the system will be laid out in such way that the reader will be able to comprehend each step of the way – from starting the process to its completion, as well as any other scenarios and conditions relating to the operations outlined.

The system section also contains a list of requirements, starting with data requirements, user requirements, environmental requirements, user requirements and usability requirements.

Following the list of requirements, we can see the architecture of the system as well an explanation of the principle how the system will work.

At the end, we can find a list of work implemented in the project so far with a brief explanation what exactly each part does.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

These are 5 of the most important requirements concerning the functionality of the system. The list is ranked from the most to the least important requirement, where the lower number indicates higher importance.

1.) The system must allow CRUD functions for tickets and users.
2.) The system must allow searching for users and tickets.
3.) The system must allow registration and login for users.
4.) The system must update the database records in real time.
5.) The system must allow specific users to create a different permission hierarchy within an organisation.

2.1.1.1. Use Case Diagram

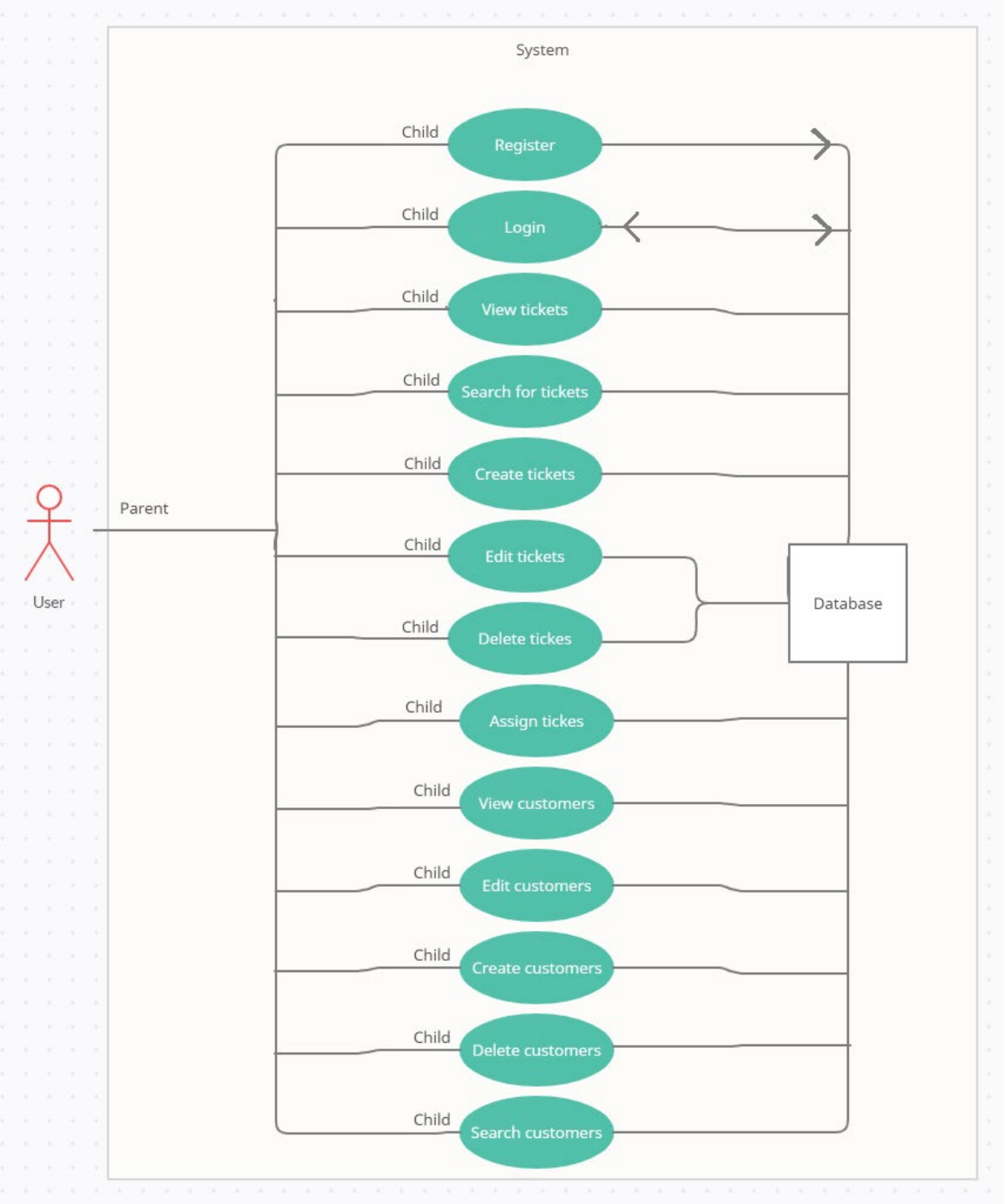


Figure 1 – Use case diagram

2.1.1.2. Requirement 1: User registration

2.1.1.3. Description & Priority

Allows the user to register. When the user registers successfully, he or she is able to start using the system.

2.1.1.4. Use Case

Scope

The scope of this use case is to register for the TeaCat System.

Description

This use case describes the registration of the end user for TeaCat System.

Use Case Diagram

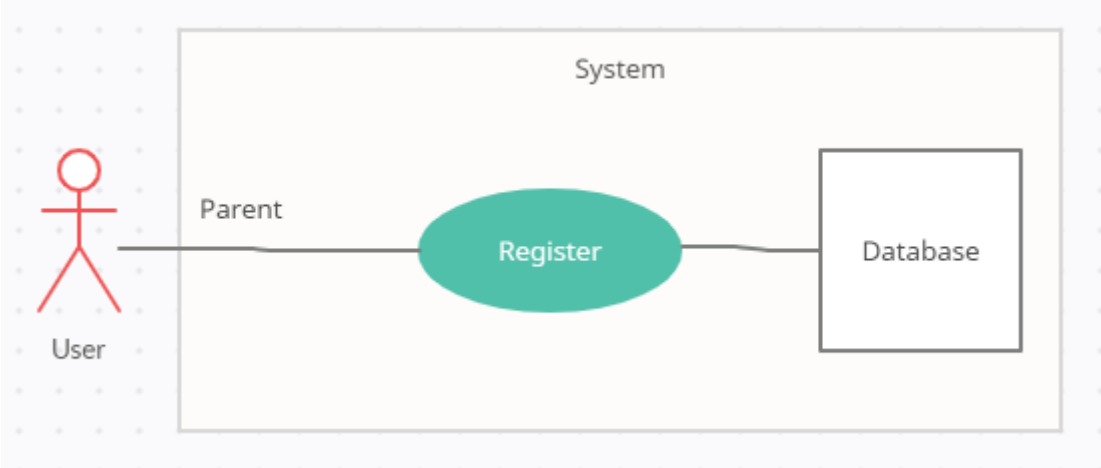


Figure 2 – Register user

Flow Description

Precondition

The system is in initialisation mode.

Activation

This use case starts when the User enters the system.

Main flow

1. The User opens a website leading to the system.
2. The User clicks on Register link (See A1 and E1).
3. The User enters their email and password.
4. The system verifies the details the user entered.
5. The system stores the details.
6. The user is brought to the home page.
7. The case ends.

Alternate flow

A1 : Login

1. The User enters his username and password.
2. The User click on login button.
3. The systems authenticates the User if credentials match the current records.
4. The User is brought to the step 6 of the main flow.

Exceptional flow

E1 : Password does not meet the criteria

1. The User enters his username and password.
2. The systems checks the password against the password rules.
3. The password is too short.
4. The User gets a message indicating what the password should include.

Termination

The system brings the User to the home screen.

Post condition

The system goes into a wait state.

2.1.1.5. Requirement 2: System Agents Operations

2.1.1.6. Description & Priority

Allows for additional Agents to be added to the system, existing Agents to be viewed, edited or deleted. The user goes into the Ticket view from where he or she can perform CRUD operations on the Agent.

2.1.1.7. Use Case

Scope

The scope of this use case is to perform the CRUD operations on internal Agents for the TeaCat System.

Description

This use case describes the possible actions on the internal Agents for TeaCat System.

Use Case Diagram

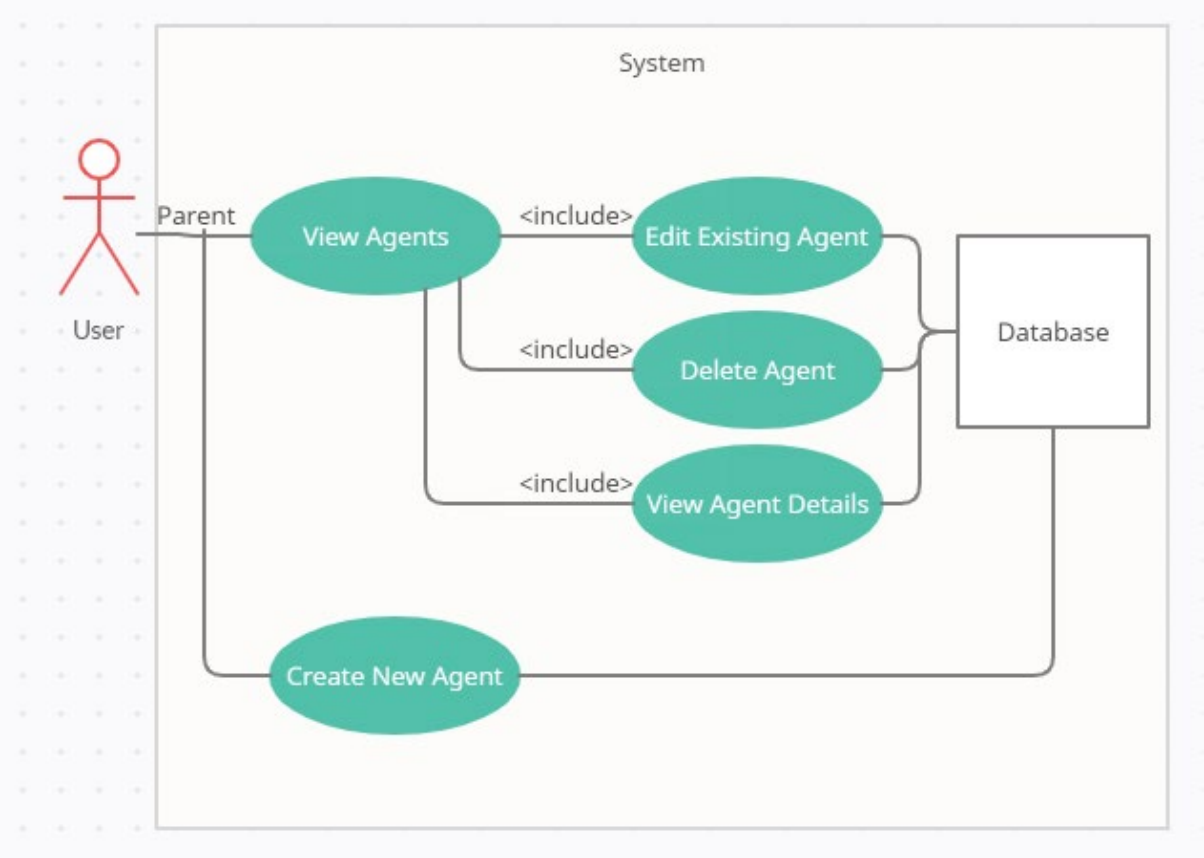


Figure 3 – CRUD operations on user Agents

Flow Description

Precondition

The system is in initialisation mode and the user is logged in.

Activation

This use case starts when the User clicks on the Agent tab.

Main flow

1. The User clicks on Agent tab.
2. The system shows the list of existing Agents.
3. The User clicks on Create New Agent (See A1 to A3 and E1).
4. The User enters name, surname and email for new Agent.
5. The system verifies the details the user entered.
6. The system stores the details.
7. The user is brought back to Agent page.
8. The case ends.

Alternate flow

A1 : View Agent Details

1. The User clicks on Details for a specific existing agent.
2. The User is presented with a page containing details about the user's name, surname, email, department and assigned tickets.
3. The user can either edit those details or go back to the previous page (See A1.1 and A1.2).

A1.1 : Edit Agent Details

1. The User makes changes to the details.
2. The User clicks on Save button.
3. The system verifies and saves the changes.
4. The user is brought back to Agent Details page.

A1.2 : Go back to previous page

1. The User click on Back to List button.
2. The user is brought back to the step 2 of the main flow.

A2 : Delete Agent

1. The User clicks on Delete button for a specific existing agent.
2. The User is presented with a confirmation message about deleting the Agent.
3. The user can either confirm this action or go back to the previous page (See A2.1 and A2.2).

A2.1 : Delete Agent

1. The User click on Delete button.
2. The user is brought back to the step 2 of the main flow.

A2.2 : Go back to previous page

3. The User click on Back to List button.
4. The user is brought back to the step 2 of the main flow.

Exceptional flow

E1 : All details are not entered

1. The User enters partial details skipping one or more fields.
2. The User clicks on Create button.
3. The User is displayed with a warning message stating that all fields are mandatory.
4. The user is brought back to the step 4 of the main flow.

Termination

The system brings the User to list of Agents.

Post condition

The system goes into a wait state.

2.1.1.8. Requirement 3: System Ticket Operations

2.1.1.9. Description & Priority

Allows for additional Tickets to be created, existing Tickets to be viewed, edited or deleted. The user goes into the Ticket view from where he or she can perform CRUD operations on the Ticket.

2.1.1.10. Use Case

Scope

The scope of this use case is to perform the CRUD operations on Tickets for the TeaCat System.

Description

This use case describes the possible actions on the Tickets for TeaCat System.

Use Case Diagram

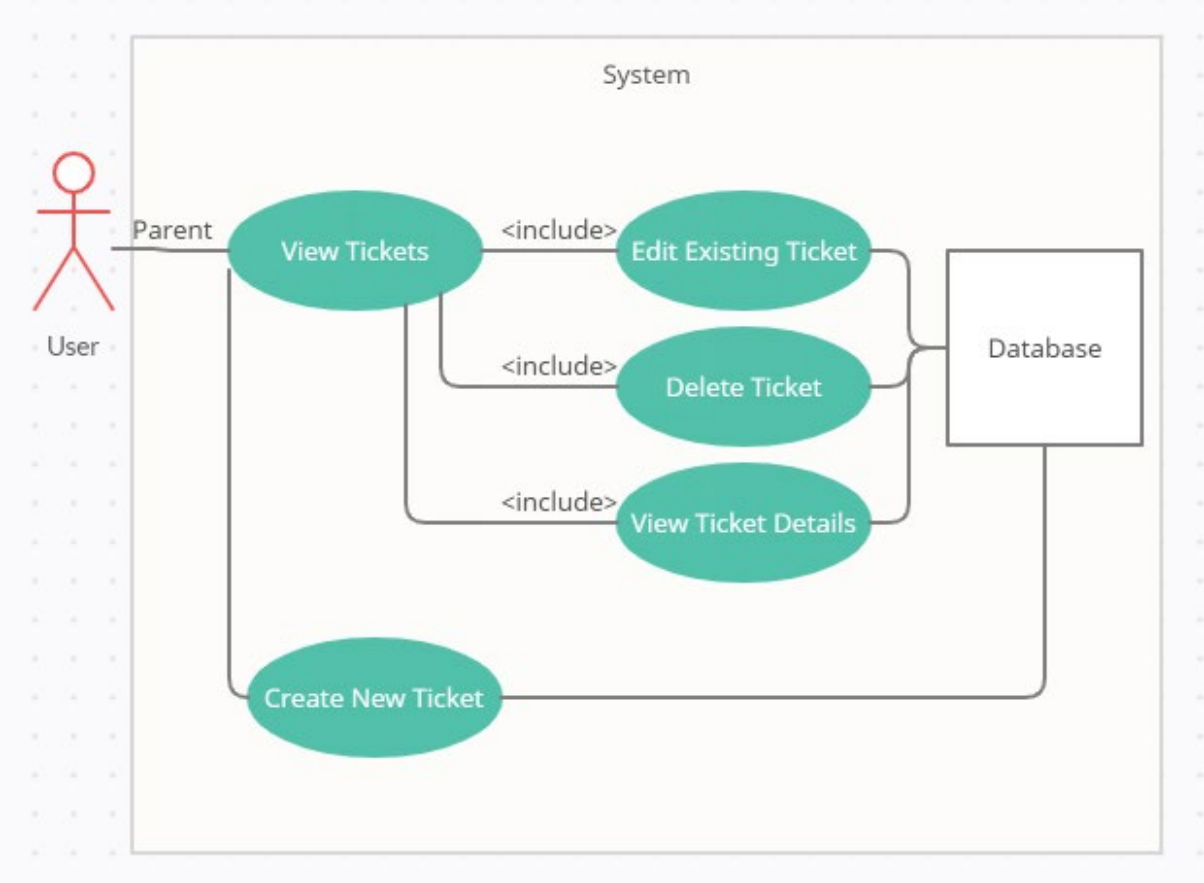


Figure 4 – CRUD operations on Tickets

Flow Description

Precondition

The system is in initialisation mode and the user is logged in.

Activation

This use case starts when the User clicks on the Ticket tab.

Main flow

1. The User clicks on Ticket tab.
2. The system shows the list of existing Tickets.
3. The User clicks on Create New Ticket (See A1 to A3 and E1).
4. The User enters Title and Body for a new Ticket.
5. The system verifies the details the user entered.
6. The system stores the details.
7. The user is brought back to Ticket page.
8. The case ends.

Alternate flow

A1 : View Ticket Details
1. The User clicks on Details for a specific existing Ticket.
2. The User is presented with a page containing details about the Ticket details, including creation time, title, body and a status.
3. The user can either edit those details or go back to the previous page (See A1.1 and A1.2).

A1.1 : Edit Ticket Details
1. The User makes changes to the details.
2. The User clicks on Save button.
3. The system verifies and saves the changes.
4. The user is brought back to Ticket Details page.

A1.2 : Go back to previous page
1. The User click on Back to List button.
2. The user is brought back to the step 2 of the main flow.

A2 : Delete Ticket
1. The User clicks on Delete button for a specific existing Ticket.
2. The User is presented with a confirmation message about deleting the Ticket.
3. The user can either confirm this action or go back to the previous page (See A2.1 and A2.2).

A2.1 : Delete Ticket
1. The User click on Delete button.
2. The user is brought back to the step 2 of the main flow.

A2.2 : Go back to previous page
1. The User click on Back to List button.
2. The user is brought back to the step 2 of the main flow.

Exceptional flow

E1 : All details are not entered

1. The User enters partial details skipping one or more fields.
2. The User clicks on Create button.
3. The User is displayed with a warning message stating that all fields are mandatory.
4. The user is brought back to the step 4 of the main flow.

Termination

The system brings the User to list of Tickets.

Post condition

The system goes into a wait state.

2.1.1.11. Requirement 4: System Ticket Search

2.1.1.12. Description & Priority

Allows for Tickets to be looked up based on specific parameters. The user enters a keyword into the search bar and clicks on the search button. After that, the system returns all records matching the search keyword.

2.1.1.13. Use Case

Scope

The scope of this use case is to look for tickets in a database based on search parameters entered by user.

Description

This use case describes the possible actions on the search function for Tickets for TeaCat System.

Use Case Diagram

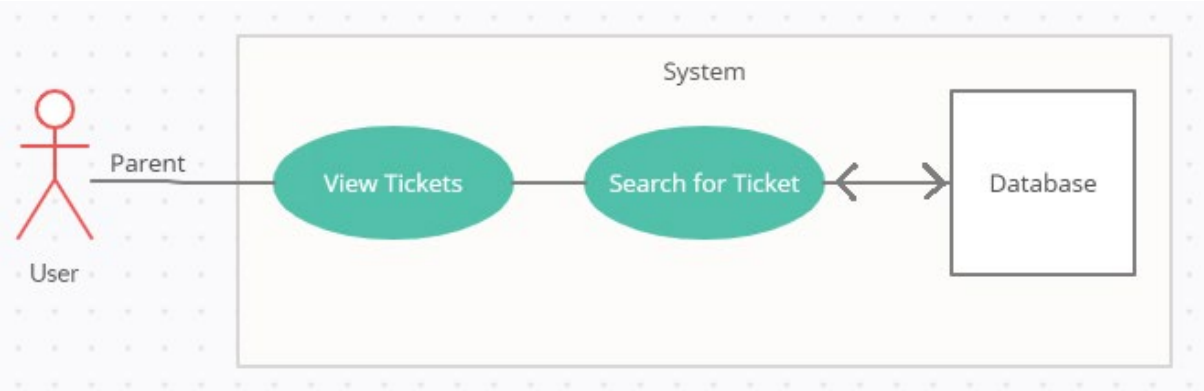


Figure 5 – Search function on Tickets

Flow Description

Precondition

The system is in initialisation mode and the user is logged in.

Activation

This use case starts when the User clicks on the Ticket tab.

Main flow

1. The User clicks on Ticket tab.
2. The User clicks on a Search field.
3. The User enters the search parameter.
4. The User clicks on Search button (See A1 and E1).
5. The system returns all tickets meeting the search criteria.
6. The case ends.

Alternate flow

A1 : No search values provided

1. The User clicks on Search button without entering any values in the search field.
2. The system returns all Tickets.
3. The case ends.

Exceptional flow

E1 : Non-existing search values are entered

1. The User clicks on Search button after entering non-existing values in the search field.
2. The system returns no results.
3. The case ends.

Termination

The system brings the User to list of Tickets matching the search criteria.

Post condition

The system goes into a wait state.

2.1.1.14. Requirement 5: System Agent Search

2.1.1.15. Description & Priority

Allows for Agents to be looked up based on specific parameters. The user enters a keyword into the search bar and clicks on the search button. After that, the system returns all records matching the search keyword.

2.1.1.16. Use Case

Scope

The scope of this use case is to look for Agents in a database based on search parameters entered by user.

Description

This use case describes the possible actions on the search function for Agents for TeaCat System.

Use Case Diagram

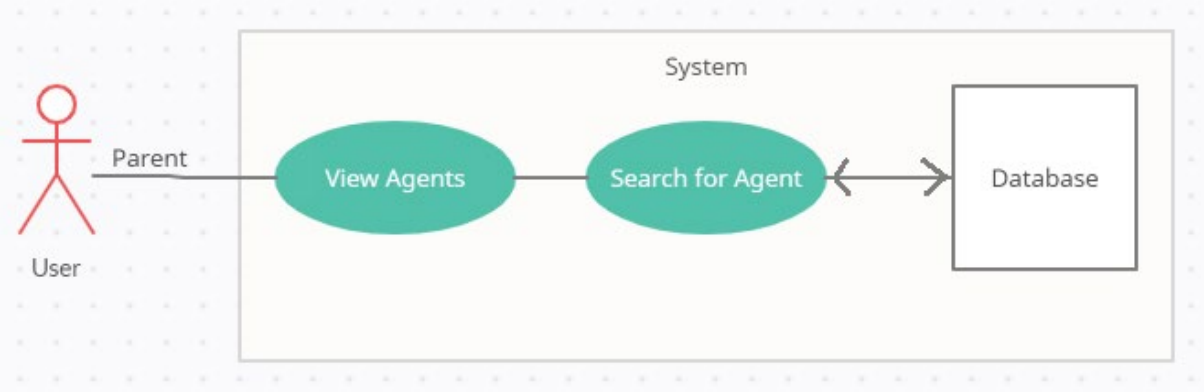


Figure 6 – Search function on Agents

Flow Description

Precondition

The system is in initialisation mode and the user is logged in.

Activation

This use case starts when the User clicks on the Agent tab.

Main flow

1. The User clicks on Agent tab.
2. The User clicks on a Search field.
3. The User enters the search parameter.
4. The User clicks on Search button (See A1 and E1).
5. The system returns all Agents meeting the search criteria.
6. The case ends.

Alternate flow

A1 : No search values provided

1. The User clicks on Search button without entering any values in the search field.
2. The system returns all Agents.
3. The case ends.

Exceptional flow

E1 : Non-existing search values are entered

1. The User clicks on Search button after entering non-existing values in the search field.
2. The system returns no results.
3. The case ends.

Termination

The system brings the User to list of Agents matching the search criteria.

Post condition

The system goes into a wait state.

2.1.2. Data Requirements

The data entered by the user will have to be stored for the future use, hence the database will be needed. The initial database used for the project is MSSQL database working off of local server. Later on, depending on any future system changes, that database might be changed to suit the business needs.

2.1.3. User Requirements

There are number of the user requirements for this project. All of them are outlined below.

1.) Security

Since the system will contain the data for users withing the organisation and the data for the users outside of the organisation, that data needs to be stored safely. In addition to that, the data cannot be accessed by anyone. As a result of that, each individual user within the organisation will have a hierarchical access to the set of permissions in order to view or modify any existing data.

2.) Availability

Since the system will be cloud based, any user within the organisation wishing to use it will have to meet some minimum requirements. Those would be stabile, reliable and moderately fast internet connection and a device from where they can access the system. Since the system will be cloud based, it will be accessible to anyone meeting these minimum conditions regardlessly of the operational system on their computer, tablet or phone.

3.) Usability

The user interface has to be intuitive and easy to navigate, leaving little to no room for any confusion from the user's perspective.

4.) Reliability

The system has to be reliable, meaning that the user can access it any time he or she wishes to do so. This has to be ensured my minimum to none downtime as well as addressing most of the common issues that could break a system. This would mean handling and checking the user's input on the front end for any form submission, ensuring that the user only enters the data that are supposed to be entered and stored in the database.

2.1.4. Environmental Requirements

The environmental factors for this system are not that numerous. The primary factor would be an active internet connection, as the system could not be accessed otherwise. If the user loses connection to the internet, he or she will effectively lose and access to the database and all of the records saved in it.

2.1.5. Usability Requirements

There are several factors which would dictate the usability requirements. They have been outlined below.

1.) Simplistic interface

If the end user has a set of repetitive tasks to complete, the last thing he or she needs is an interface that is filled with dozens of instances of features the user is not using and most likely will never use. The focus should be on displaying only a simplistic view which is relevant for the user, therefore minimising the chances for any confusion and making the whole interaction straightforward.

2.) Intuitive design

The system should be designed in such way that the progression from one activity to the other comes naturally without the end user having to put too much thought into where to go next and what to do. As most of our experience in using various technologies and platforms comes from our previous experience, it would be important to make some logical design decisions that would not confuse the user, like not putting the menu on the bottom of the page.

3.) Security

As there will be organisations of different sizes with different hierarchies involved, it would be important that their details are stored securely within the system. Also, users should have a special set of the account permissions, limiting what data are they allowed to access within the system.

2.2. Design & Architecture

The TeaCat System was developed using the MVC 5 to address the separation of concerns. In this model, we have 4 main entities – Models that define the data structures, Controllers - which act as a mediator between Models and Views as well as perform any operation in the system, Views – essentially pages displayed for the end user and a database – where all records are stored. The architecture of the MVC model is displayed below in the figure 7.

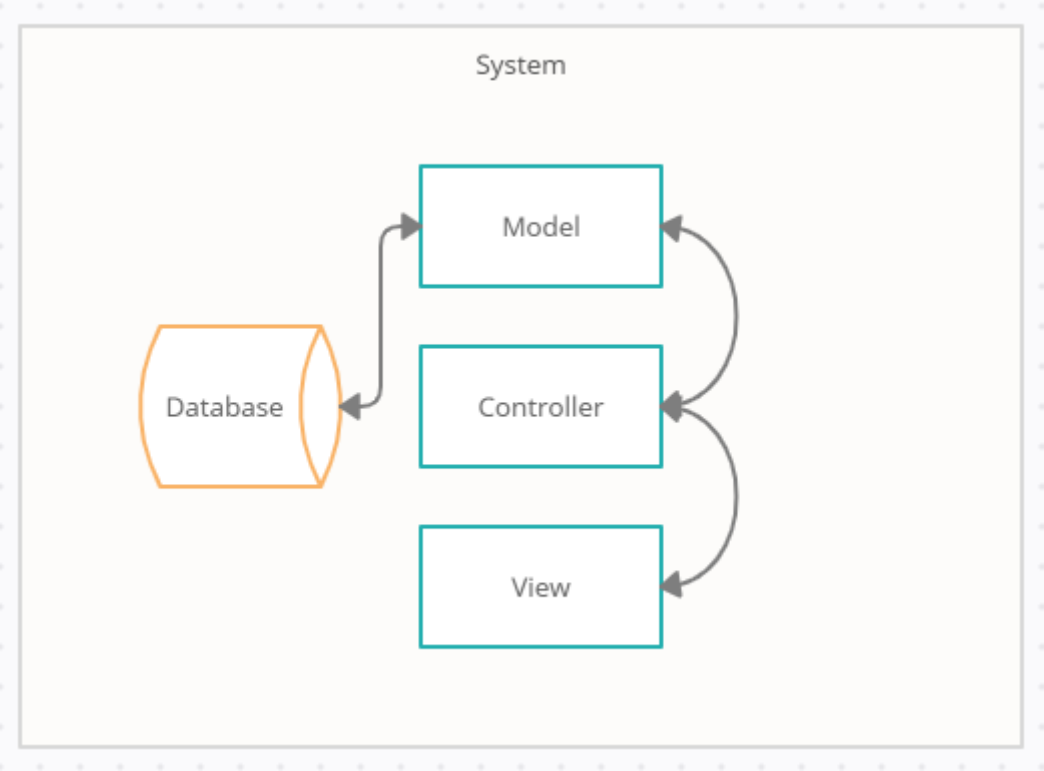


Figure 7 – The MVC model

2.3. Implementation

So far, the project contains a number of Models defining data structures:

- Agent model – defining attributes for the Agent role in the system
- Ticket model - defining attributes for the Ticket structure in the system
- Assignment model – representing connection between the Agent and Ticket
- AccountViewModels – defining attributes for user
- ManageViewModels – defining attributes surrounding user login details

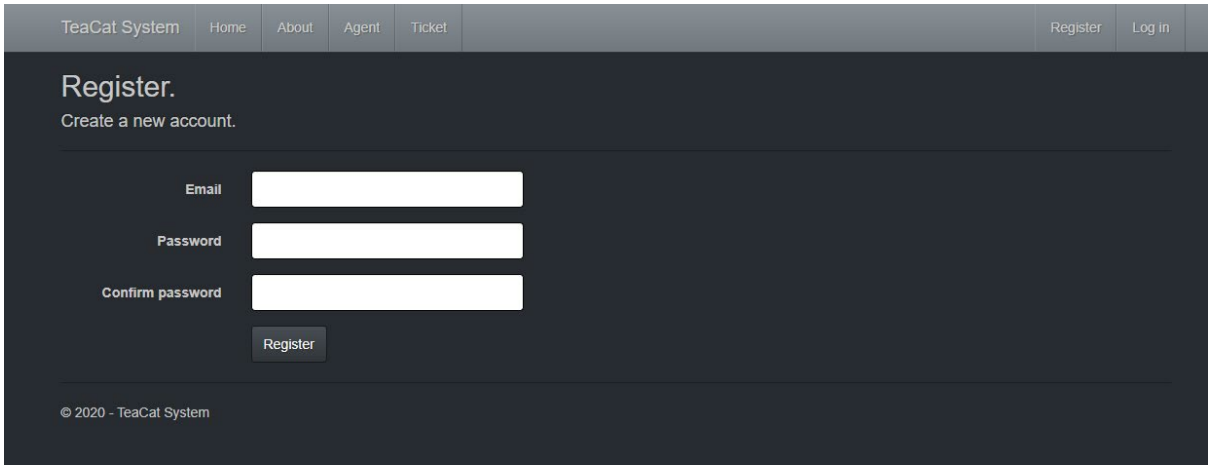
Aside from Models, Controllers have been built as well:

- Agent controller – providing CRUD and search functions for Agents
- Ticket controller - providing CRUD and search functions for Tickets
- Home controller – providing change between views on the home page
- Manage controller – operations relating to user accounts
- Account controller – operations regarding the user login

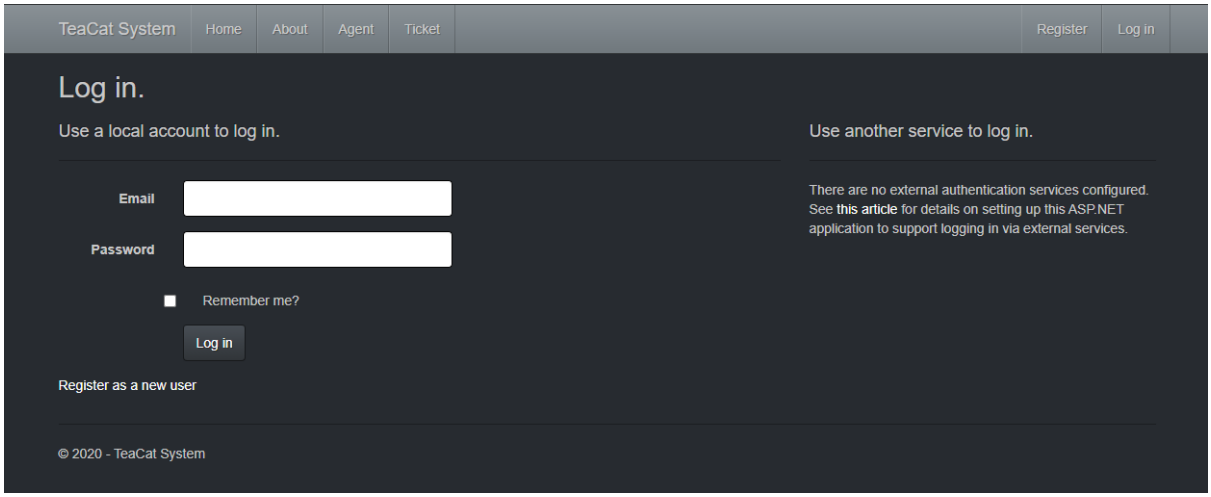
Finally, Views are as follows:

- Agent (Index, Create, Delete, Edit, Details) – various views concerning Agent
- Ticket (Index, Create, Delete, Edit, Details) – various views concerning Ticket
- Home (Index, About) - various views concerning navigation from Home page
- Account (Login, Register, ResetPassword etc.) - various views concerning the User

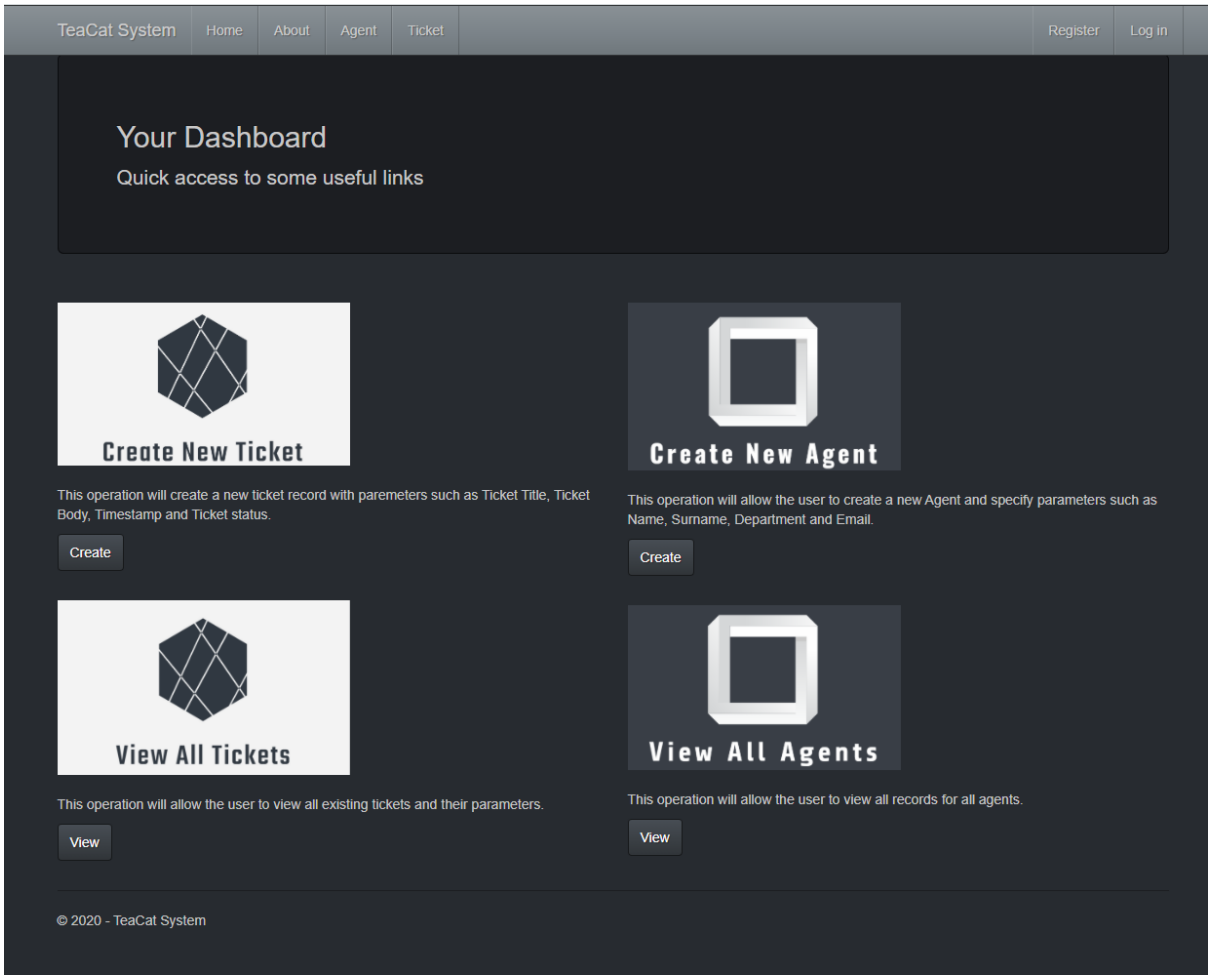
2.4. Graphical User Interface (GUI)



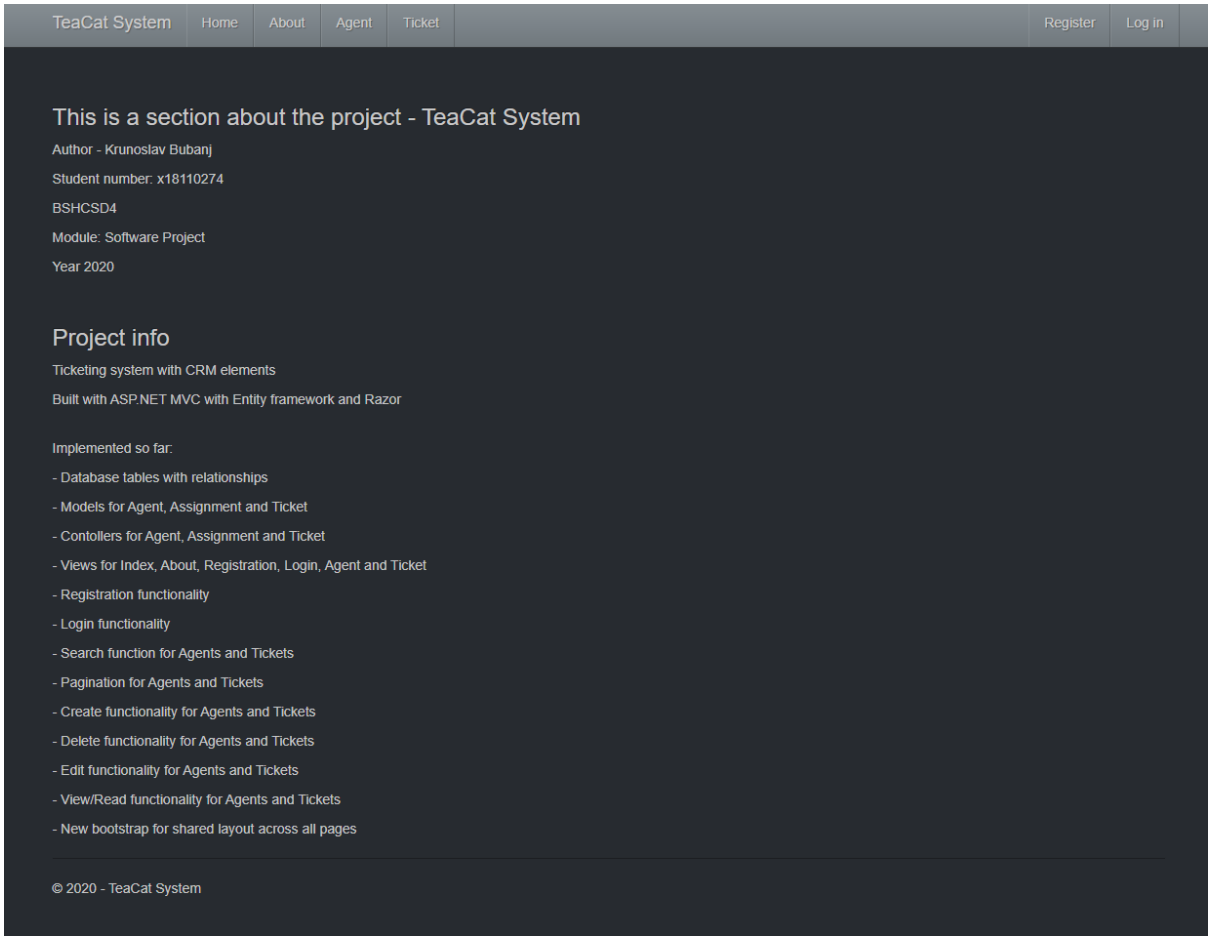
Register – user enters his or her details in order to register.



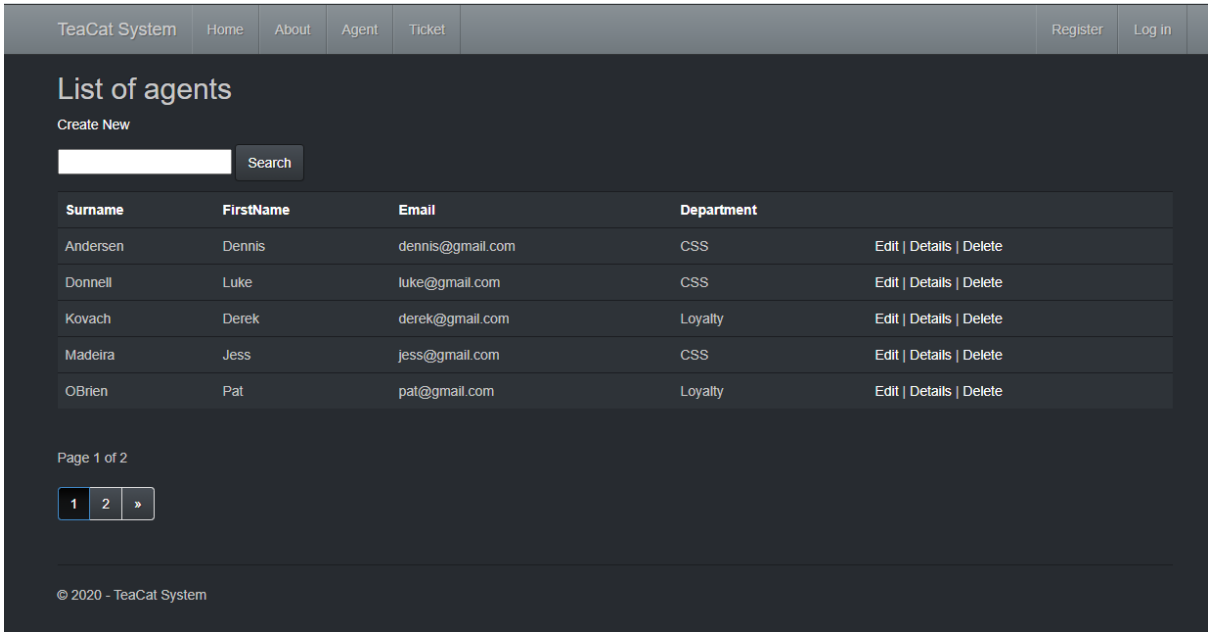
Login – user enters the login details in order to access the system.



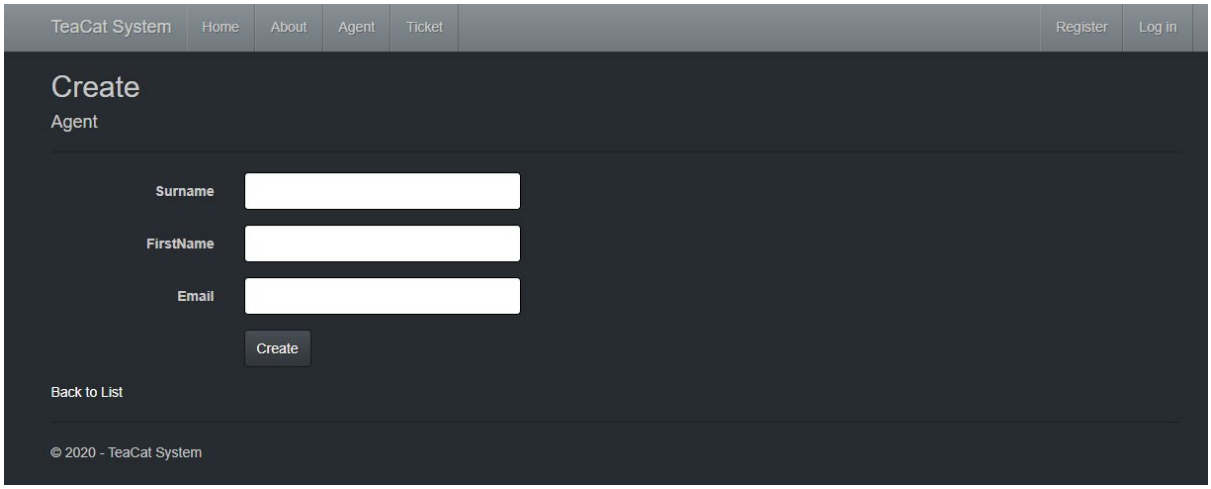
Index page – contains nav bar with other parts of the system as well as quick links for viewing and creation of agents and tickets.



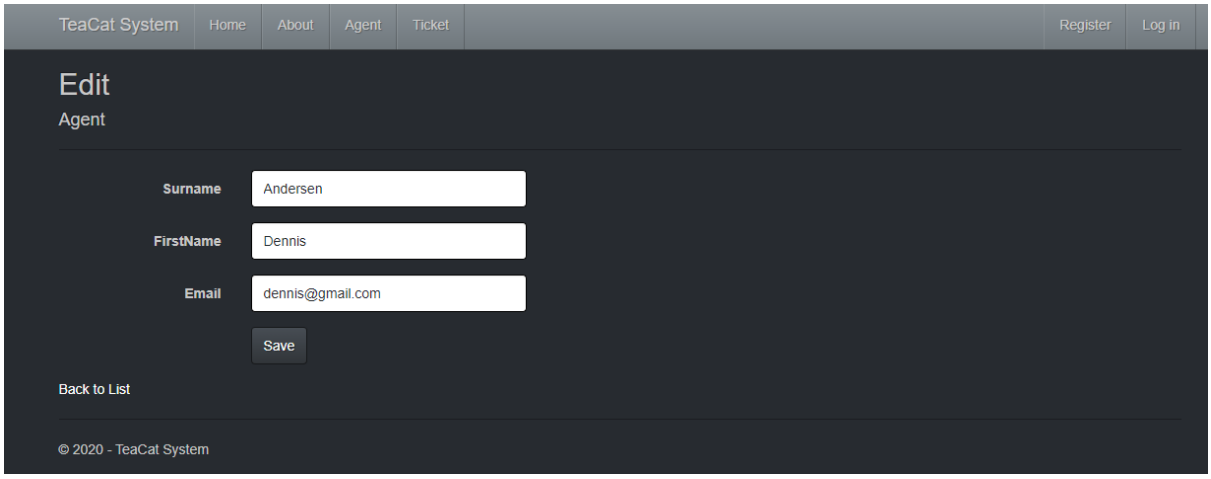
About page – indicates the progress so far as acts as a placeholder.



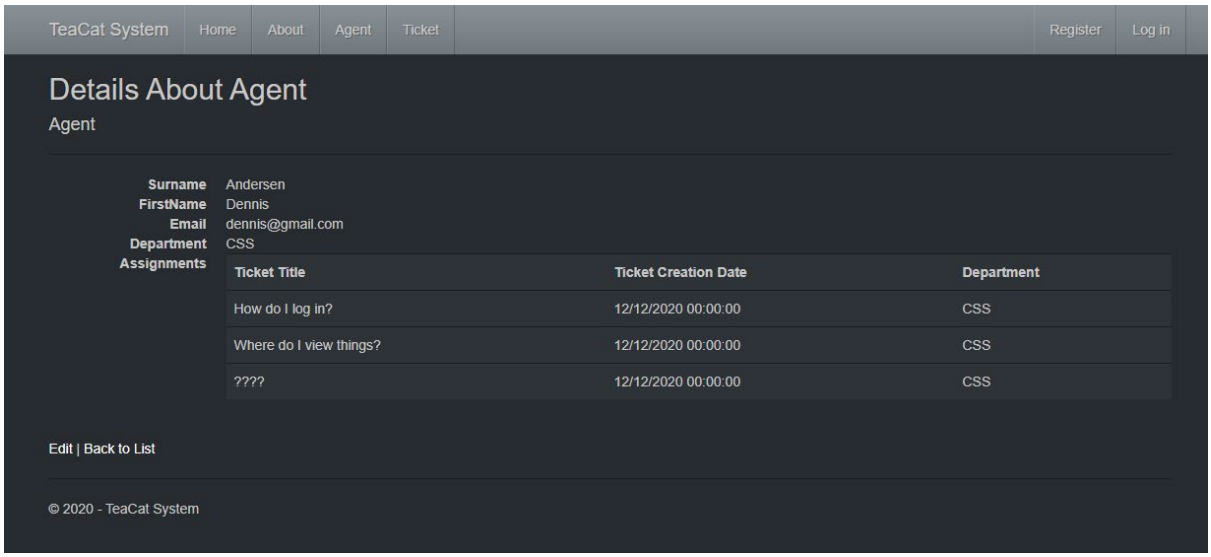
Agent index page – records of all agents in the system.



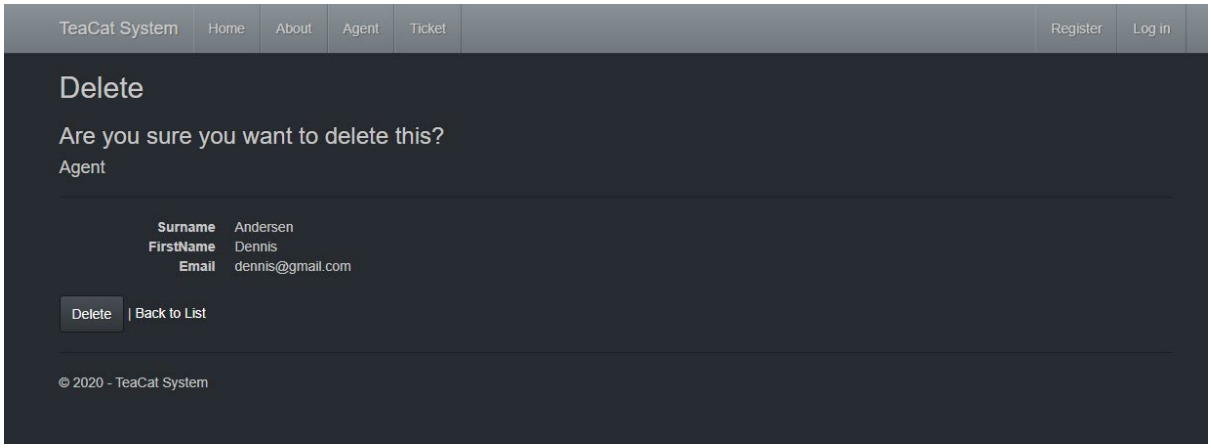
Create new agent page – adds a new agent to the database.



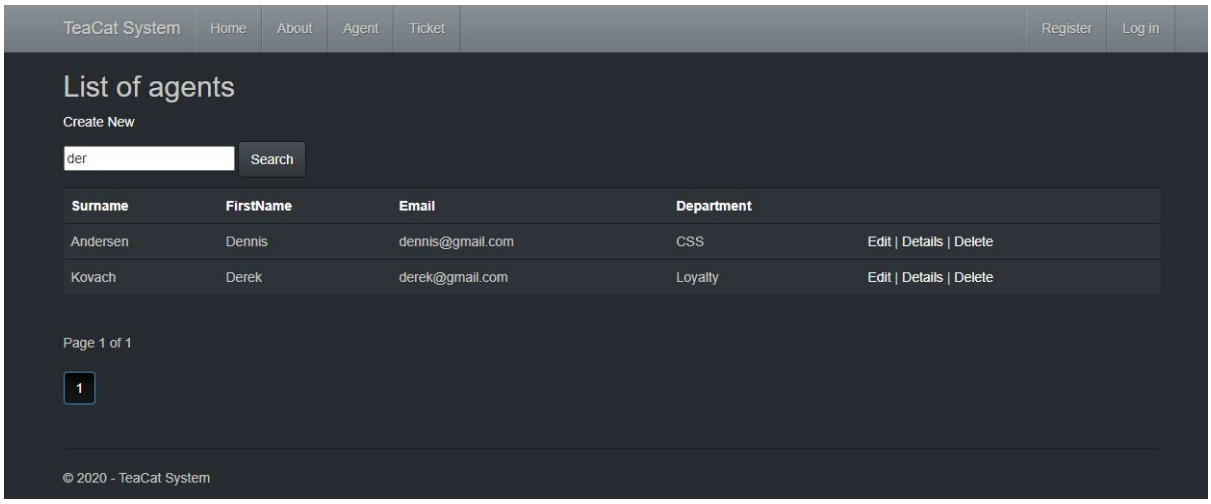
Edit agent page – edit details from an existing agent.



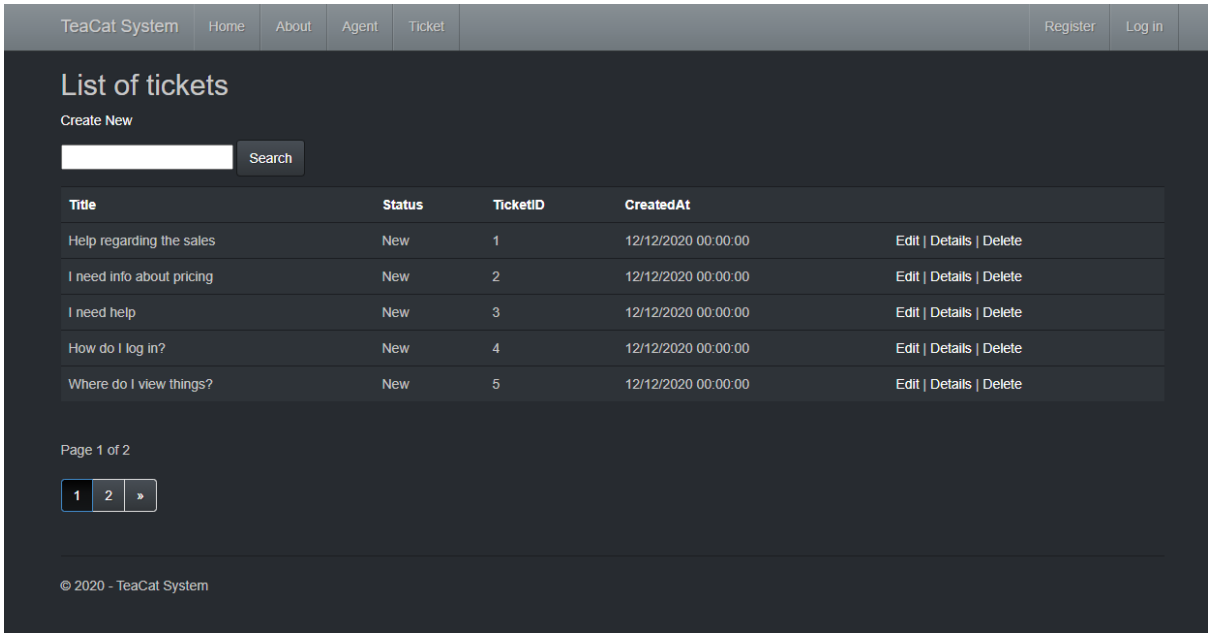
Agent details page – information about the agent and the work assigned to him or her.



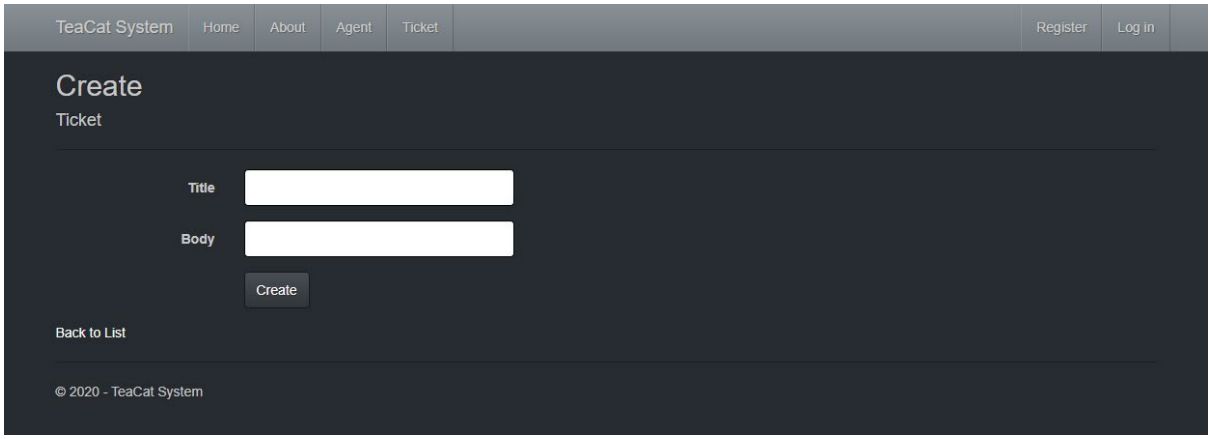
Delete agent page – deletes and existing agent record from database.



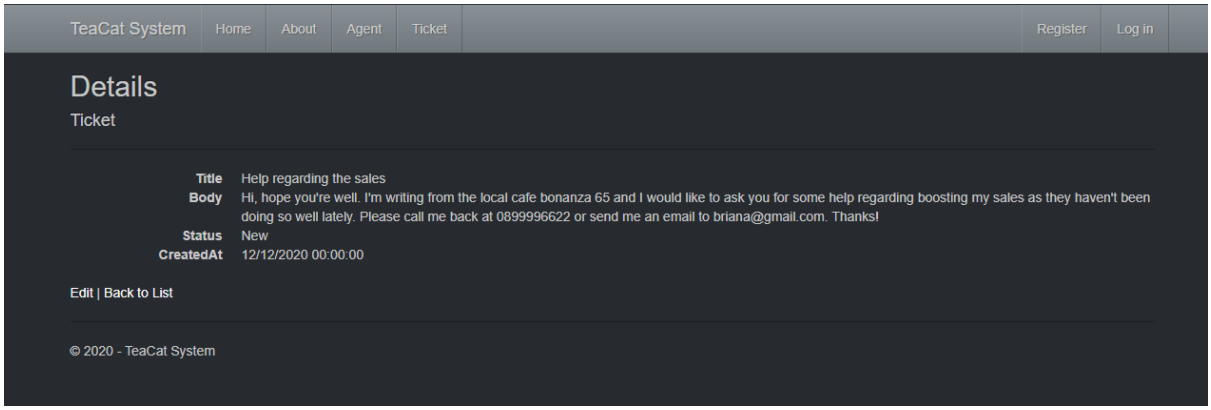
Agent index page – search function lookup for agent records.



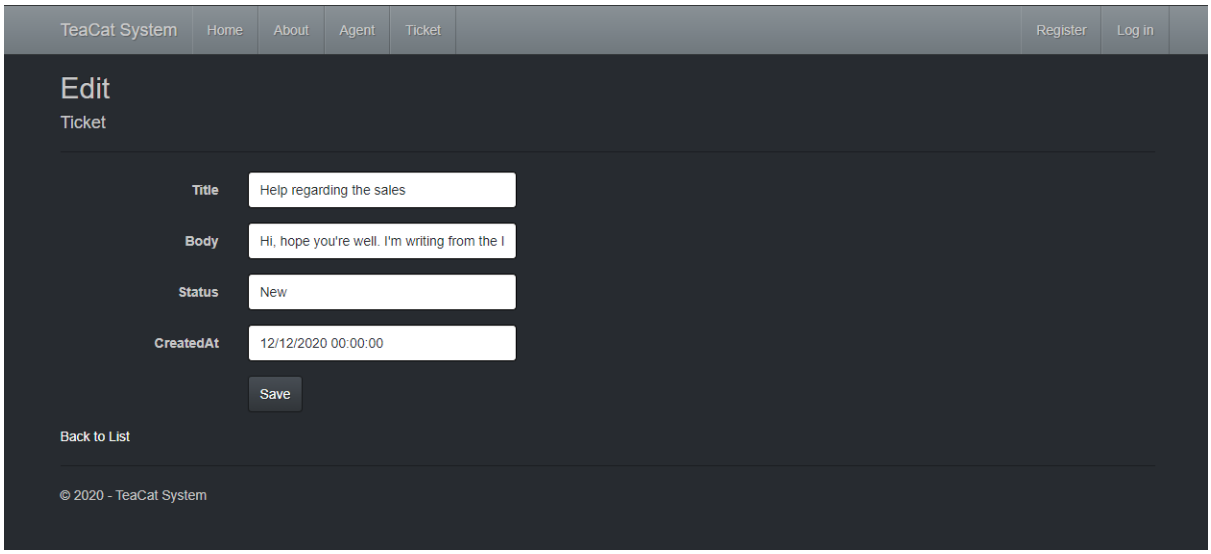
Tickets index page – information about all existing tickets.



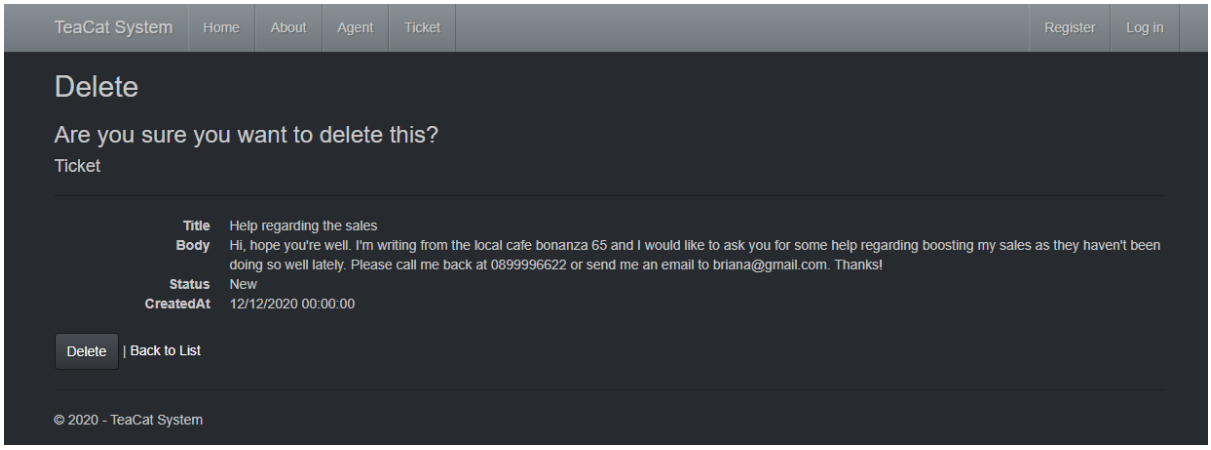
Create ticket page – creates a new ticket record.



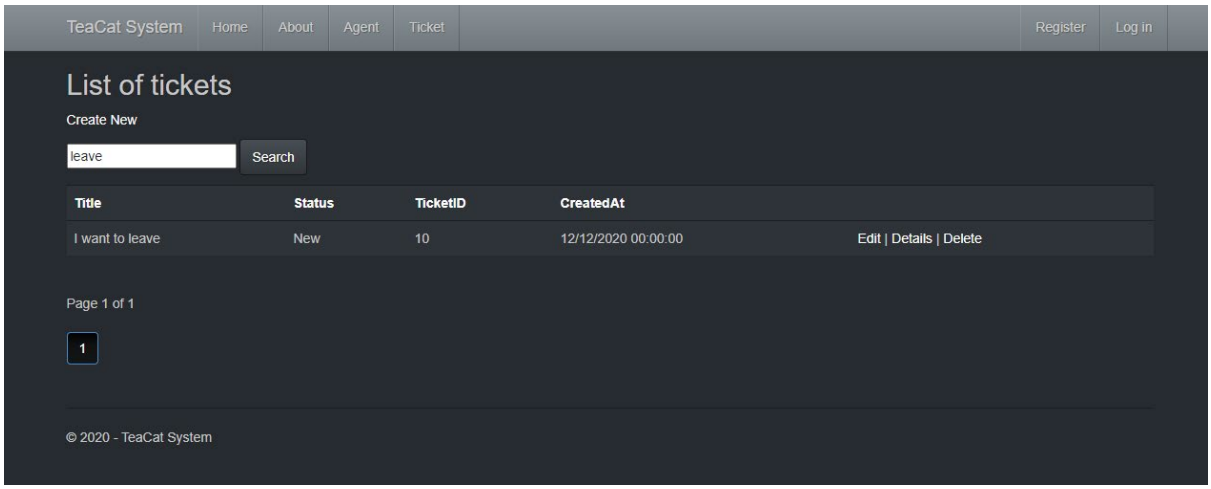
Ticket details page – shows details for an existing ticket.



Edit ticket page – edit the details for an existing ticket.



Delete ticket page – delete a record of an existing ticket from database.



Search ticket page – finds all ticket records matching the search parameters.

2.5. Testing

Describe any testing tools, test plans and test specifications used in the project. Provide evidence for and results of all Unit, Integration and End User testing that is carried out.

2.6. Evaluation

How was the system evaluated and what are the results? This may consist of usage data. It may also include performance evaluations, scalability, correctness, etc. depending on the focus of the project. Quantative results may be reported in tables or figures.

3.0 Conclusions

Describe the advantages/disadvantages, strengths and limitations of the project

4.0 Further Development or Research

With additional time and resources, which direction would this project take?

5.0 References

Please include references throughout your document where appropriate. See here for a guide on referencing from the NCI library.

6.0 Appendices

Appendix 1 - Project proposal

1.0 Objectives

With the current state of affairs in the world caused by pandemic, it is becoming more and more evident that most of the businesses are moving their trading partially or fully online in order to adjust to the current climate and survive through it. Since online trading inevitably calls for system upgrades and updates like different payment system implementation to cater for the majority of the people today, different and new platforms coming out every year, various new features, progress tracking, advertisement on different platforms and other aspects, new laws and legislations that all companies need to abide to, it is reasonable to assume that businesses will need an efficient way to cope with all new changes and manage all of their new business aspects efficiently.

If we take an average restaurant as an example, we can clearly see that their classic way of conducting business within their premisses in no longer achievable. Constantly changing government guidelines and enforcement of various lockdown levels have thrown a classical restaurant to its knees. If the restaurant does not change their modus operandi to adjust to the new business environment, it is destined to fail.

However, if the restaurant manages to adjust and switch their business online, they will most likely need to partner up with a company like Just Eat, Doordash, Grubhub, Flipdish or a similar company. Those companies are companies that will switch the business type from in person to online. In this year, it would be reasonable to expect that those companies are seeing a massive growth in the number of their clients. It would also be reasonable to assume that due to their growth, they are hiring more people to meet the growing demand. With the sudden increase of staff members, it would be reasonable to assume they will need a system to keep track of the various aspects of the requests sent in by their clients, whether it is a complaint, feature request, bug report or something else.

TeaCat system is a system that will try to take all of the assumptions, validate them from the relevant field, take the results into consideration and make a system that will incorporate Client Relation Management system with a Ticketing system to create an entity which will address all issues with the current systems most of the companies are using. It will aim to automate most of the processes that are done manually like ticket assignment, GDPR compliance check, extracting customer data manually and overall – save the time needed to perform those tasks and allow the staff to spend their time more productively. The

restaurant business model was just one example in the ever-growing list of companies conducting their business online.

2.0 Background

The idea behind the TeaCat system came from my personal work experience. So far, I have worked in various positions in different companies and different roles, whether it was a technical support role, sales role, retail role or a customer support role and regardless whether it was a huge corporation, medium size company or a start up company, they all seem to have an issue where they are using several different systems to accomplish certain tasks.

In the current company I work for, we are using over five different software solutions to accommodate the need for saving information about incidents, customers and sales. The part that worries me the most is the fact that each system in use is creating charges per person using it, which would mean that companies with over 100 employees have a hefty amount to pay each and every month not just for one, but for each and every system they are using. When speaking with other people in the company, they also seem to agree that this approach is not the greatest and programs we are using are not fulfilling all needs, but developers are too busy to start working on a in-house built solution and other software solutions that would solve some of the issues our current systems have just seems to be too expensive. With everything taken into consideration, the general census in the company is that everyone is able to work with the current set of systems and overlook the limitations since introducing other available solutions is deemed to be too expensive.

If I am going to look even further back in my employment history with one of the biggest Irish internet service providers, they were facing pretty similar issue. However, their management board decided to build their own ticketing system. While it did have a positive effect on the overall performance of the staff members, they still had to use other systems for GDPR and DPA verification purposes, sales and retention lead documents and technical support logs, tasks and documentation. Since today most of the similar systems in use are mostly cloud based, companies are no longer dependant of obtaining and maintaining their servers which in return means savings on the cost of network engineers and hardware technicians, infrastructure costs, back up security and other related costs. With that in mind, it seems to me highly illogical that companies would rather operate on multiple different platforms that are not well connected or connected at all with each other compared to having a one gateway system which is scalable and addresses all of the business needs.

I have even noticed that National College of Ireland uses Zendesk for their technical support helpdesk. Since I have a several years of experience working with Zendesk, I can come to a conclusion that Zendesk has certain limitations like limited overview, significant lack of access levels and user roles that would accommodate medium and large size companies,

inability to customise the ticket view to the full extent of user needs, inaccurate reporting issues, lack of knowledge base, limited ticket states and additional useful features locked behind a paywall.

By thinking about this issue from a technical point of view, the entire concept simply comes down to a database with added functions and database queries simplified and adjusted for all user profiles though intuitive UI.

The aim of TeaCat system would be to provide a highly customisable user interface where its users can adjust their search parameters, extend database table if needed with new columns, have a way to build their own queries to extract only the data that is considered to be relevant to them and provide a better overview of the active tickets.

The main idea would be to have end users registered in a system. That way, when that particular end user sends in email, TeaCat system would find the user in records, pull out all relevant data to that user and automatically assign or forward that ticket to the person or a department in charge of that end user.

The internal user groups would be introduced hierarchically where their roles and access levels could be customised to cater for the needs of any newly created departments or roles.

The TeaCat system should also need to be scalable to ensure it covers all company sizes as well as almost certain future feature requests to be implemented.

Another thing to keep in mind would be the ability to integrate the TeaCat system in other third party systems like Jira, Slack, Pipedrive, Salesforce and others.

3.0 Technical Approach

The project will be completed using the waterfall methodology as all stages are clearly defined already.

The system will require the user to be able to register, login, customise the interface, capture data, add records, delete records, search for records, write custom queries for data extraction, automation of simple processes (where possible), create reports, set various role

permissions within an organisation and to be able to communicate with people outside of an organisation.

All these features and functionalities are planned to be implemented into the final product with the usage of C#, SQL and ASP.NET MVC 5 framework.

Some companies that have similar products I have looked into are Zendesk, Jira and Salesforce.

Zendesk drawbacks – lacking the ability to customise the user interface, inaccurate reporting system and a bad overview of all tickets. Tends to be quite expensive for the level of the functionality it provides.

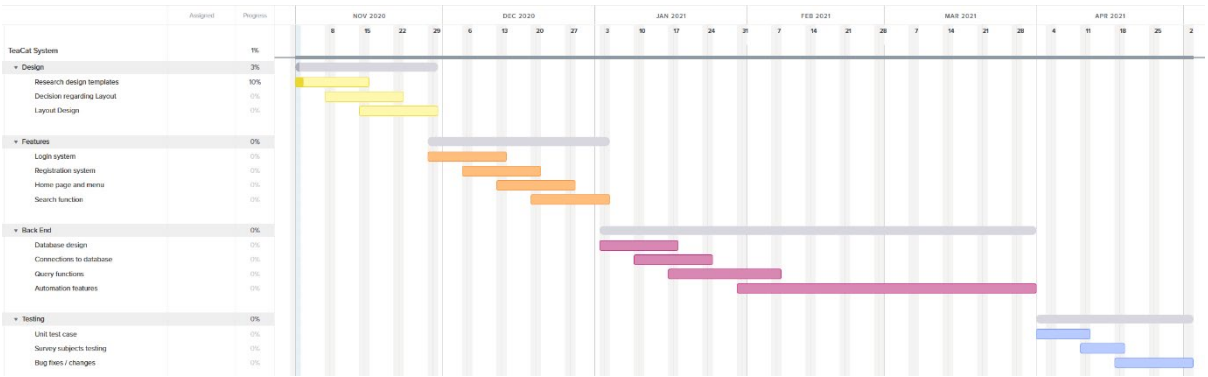
Jira drawbacks – aimed at developers and project managers with minimum to no customisation of the user interface. No way of capturing reusable data.

Salesforce drawbacks – covering most of the features but way too expensive for a typical company with costs only rising as features are being added to the existing plan.

4.0 Special Resources Required

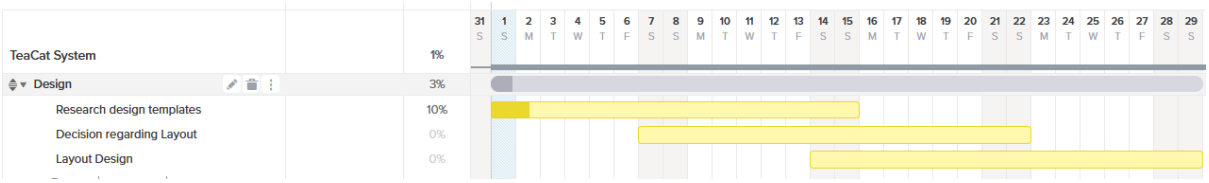
The TeaCat system will not require special resources, as only a personal computer with a decent internet connection above 5Mbps should be enough to use it.

5.0 Project Plan



https://prod.teamgantt.com/gantt/schedule/?ids=2389496#&ids=2389496&user=&custom=&company=&hide_completed=false&date_filter=&color_filter=

Phase 1 – Design



In this phase there are 3 parts I wish to complete:

a) Research design templates

This will give me a better idea of what design works well with all functionalities I had in mind. The idea is to go over various competitors' platforms and random design templates to determine which one can work quite well. This part is scheduled to start at 1st November and finish by the 15th November.

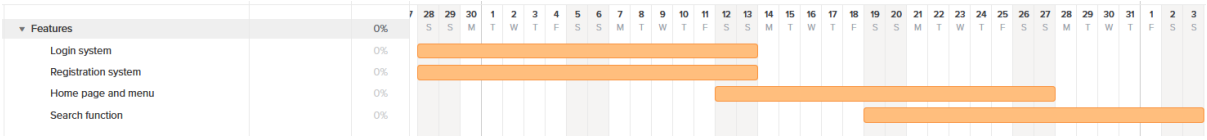
b) Decision regarding layout

This stage will include narrowing down the appropriate layout templates and colour schemes to be used and making a final decision. This part is scheduled to start on the 7th November and to be completed before 22nd November.

c) Layout design

With the initial research completed and additional knowledge acquired, I should be able to make an efficient layout design for the TeaCat system. This part is set to begin on 14th of November and to be completed by 29th November this year.

Phase 2 – Features



a) Login system

Once the design idea is "set in stone", it is time to move on to the functional parts of the software. The first part would be the login system which is scheduled to start on the 28th of November and to be completed by 13th of December.

b) Registration system

It only makes sense that the registration system is developed in parallel with login system. I divided those two tasks as the registration has to be broken down into several layers like organizational, department, managers and staff. It is also scheduled to start on the 28th of November and to be completed by 13th of December in order to be synchronised with login.

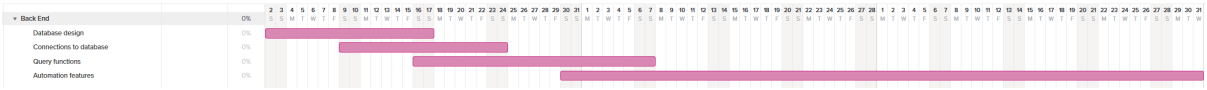
c) Home page and menu development

The home page is planned to act as a main user interface from where the user can get an access to other systems and perform CRUD operations. The development of the main interface aspects is scheduled to being on the 12th of December and to be completed by the 27th of December.

d) Search function

The search function will enable user to quickly search over the records using some parameters but only allowing him or her an access to the files their roles is set to. Since I am counting to run unto some security issues at this stage, this feature alone is scheduled to start on the 19th of December and to be completed by the 3rd of January next year.

Phase 3 – Back end



a) Database design

In this phase I will try to design all database tables while including parameters which I believe will be used universally and making it scalable. This part is scheduled to start on the 2nd of January and to be completed by the 17th of January.

b) Connection to database

Since the software will have to update the records in near real time, I will try to make the connections as secure and fast as possible to avoid any potential database concurrency issues. This is scheduled to take place from the 9th of January to the 24th of January.

c) Query functions

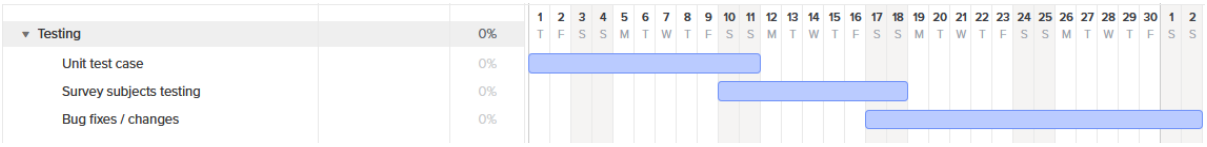
As discussed previously, my aim is to allow users to perform a wide spectrum of database queries with minimum to no knowledge about programming. I have a rough idea how to achieve this but it will still need a good bit of polishing in order to be implemented successfully. This part is scheduled to start on the 16th of January and to be completed by 7th of February.

d) Automation features

The last part of the back end development will be various automation processes. As indicated previously, I intend to automate as many processes as possible, which would include ticket assignment, pulling up clients' data, service level agreement notification,

weekly reports and anything else that I deem could be automated. This will by far be the hardest and lengthiest part of the project, hence the timeframe set for this is from 30th of January until the 1st of April.

Phase 4 – testing



a) Unit test case

In this stage the finished software will be tested for any bugs or any potential issues that may surface later. This stage is scheduled to commence on the 1st of April and to be completed by the 11th of April.

b) Survey subject testing

This is the stage where human participants will test the software on a voluntarily basis. Upon successful completion of the testing period, they will be asked to share their opinions via survey to determine any possible issues and drawbacks of the system. Testing is scheduled to start on the 10th of April and to be completed by the 18th of April.

c) Bug fixes and changes

This stage comes as a direct result from the previous stage of testing. A period of 2 weeks is set in case some major bugs are discovered and need to be addressed before the project is finalised. It is scheduled to last from 17th of April until the 2nd of May, bringing the project to conclusion.

6.0 Technical Details

The plan is to develop the TeaCat system using C# programming language in conjunction with SQL for querying databases.

To go into a bit more details, the aim will be to develop it using ASP.NET framework with MVC architecture.

Since I only had a brief exposure to C#, I find it challenging, yet rewarding to attempt and complete this project using ASP.NET. I am planning to use some of its native libraries to

accomplish user roles and permissions as well as login and registration system, while the rest of the search functions, data creation and storage, views and design will have to be manually coded.

7.0 Evaluation

The product will be evaluated in several stages:

- Fragmental testing during development
- Testing of the finished product
- End user testing

Fragmental testing, as the name suggest, will be done in fragments as the project progresses. Each new feature will be tested in the development environment and determined whether it behaves as expected or no. Each fragment represents a feature implementation.

Testing of the finished product will occur once all stages of the project development are completed, all bugs addressed and the final product is deemed to be ready for further testing. In this stage the product will be tested on different browsers and different platforms to ensure compatibility with other systems and environments as well as all feature functionality.

The final stage is an end user testing where people will test the product on a voluntarily basis. This stage is set to last up to one week after which end users will be asked to fill an anonymous online survey in relation to the product functionality, new feature wishes and suggestions and overall concerns they may have. Their answers will be collected and analysed while their personal details will not be stored anywhere.

The data from all 3 stages will be collected to determine what will be feasible in relation to time left for the project completion, where any functional shortcomings and bugs will take priority over the new feature request implementation.

7.1. Project Plan

7.2. Ethics Approval Application (only if required)

7.3. Reflective Journals

7.4. Other materials used

Any other reference material used in the project for example evaluation surveys etc.