National College of Ireland

BSc in Computing

2017/2018

Khateeb Ahmad

X16112989

x16112989@student.ncirl.ie

Final Project Report

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: **Khateeb Ahmad**
Student ID: **X16112989**
Supervisor: **Dr. Paul Stynes**

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Khateeb Ahmad_____ Date: 13 May 2018

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Table of Contents

Executive Summary

Problem

Finding the right service at the right time is an extremely challenging task currently. Our own experiences and surveys that we have conducted, indicate that it is a pain-point and needs immediate attention.

Solution

Such consumer problems can be solved by leveraging power of the internet and creating a matchmaking engine which sets up potential customers with the quality service providers in their area, saving them from the hassle of going through thousands of listings or having to rely on word-of-mouth referrals.

1 Introduction

1.1 Background and Aims

Managing too many tasks can be a real challenge, but using technology based solutions can make it easier. EazyServices is an application that aims to be a one stop solution for all routine service dependencies. The application offers users a wide range of services such as Repair & Maintenance, Event Planning, Health & Wellness and other services to make it a complete lifestyle platform. To use EazyServices, a customer selects a category and answers several questions on website. Our platform's recommendation engine uses the information and location-based data to match users with service providers. Application allows users to book appointments instantly in a hassle-free manner. It also allows the users to have a direct chat with the professionals/service providers. Through EazyServices, I am planning to build the largest web based services marketplace in Ireland. Professionals can join to offer their services on-line and are rated on a five-star scale by customers, which helps them maintain quality. EazyServices can essentially be a one stop destination for all the professionals to expand their market on-line. A list of services that would be included are:

- Location-based search for service providers
- Rating and Reviews to be given by users to service providers. A sentiment analysis would be done on each review using the semantic web to look for positive or negative sentiment and rate the service provider accordingly.

- Facebook integration to sort reviews based on Facebook friends
- Robotic process automation to ease the task of finding service providers for users.
- Book an appointment functionality to allow homeowners to avail services provided by service providers.
- Payments functionality to allow customers to pay service providers from the portal itself.
- Coupons for customers to get a discount on the bill for services taken and Rewards for service providers for providing quality services.
- Monetization - the app charges a small percentage of fees from the customer/service provider for the platform provided. This will help development team make money.

1.2 Technologies

EazyServices solution will be accessible using a web-based graphical user interfaces. The interface will be developed using HTML5 technology. Clients will be RESTful and use HTTP standard methods to communicate with middle tier services.

Middle tier of EazyServices will consist of many micro services instead of traditional monolithic web application. Micro services are scalable, heterogeneous, provides clear contextual boundaries and ease of independent development and deployment. It is relatively easy to distribute and fault tolerant due to its self-contained nature

and concise specification. I will use Java Standard Edition and Enterprise Edition to develop micro services.

To store business, configuration and subscription data, I will use relational databases. Database will have secured access only to authorized administrator. The micro services will authenticate all queries before executing them on databases.

I will use an object relational mapping tool such as Hibernate to associate data with objects on middle tier. I will also take care of major cross cutting concerns such as logging, authentication, authorization, communication, management and configuration across all tiers of application, wherever applicable. Industry standard Spring Framework and its aspect oriented capabilities will be used in middle tier to handle cross cutting concerns.

1.3 Structure

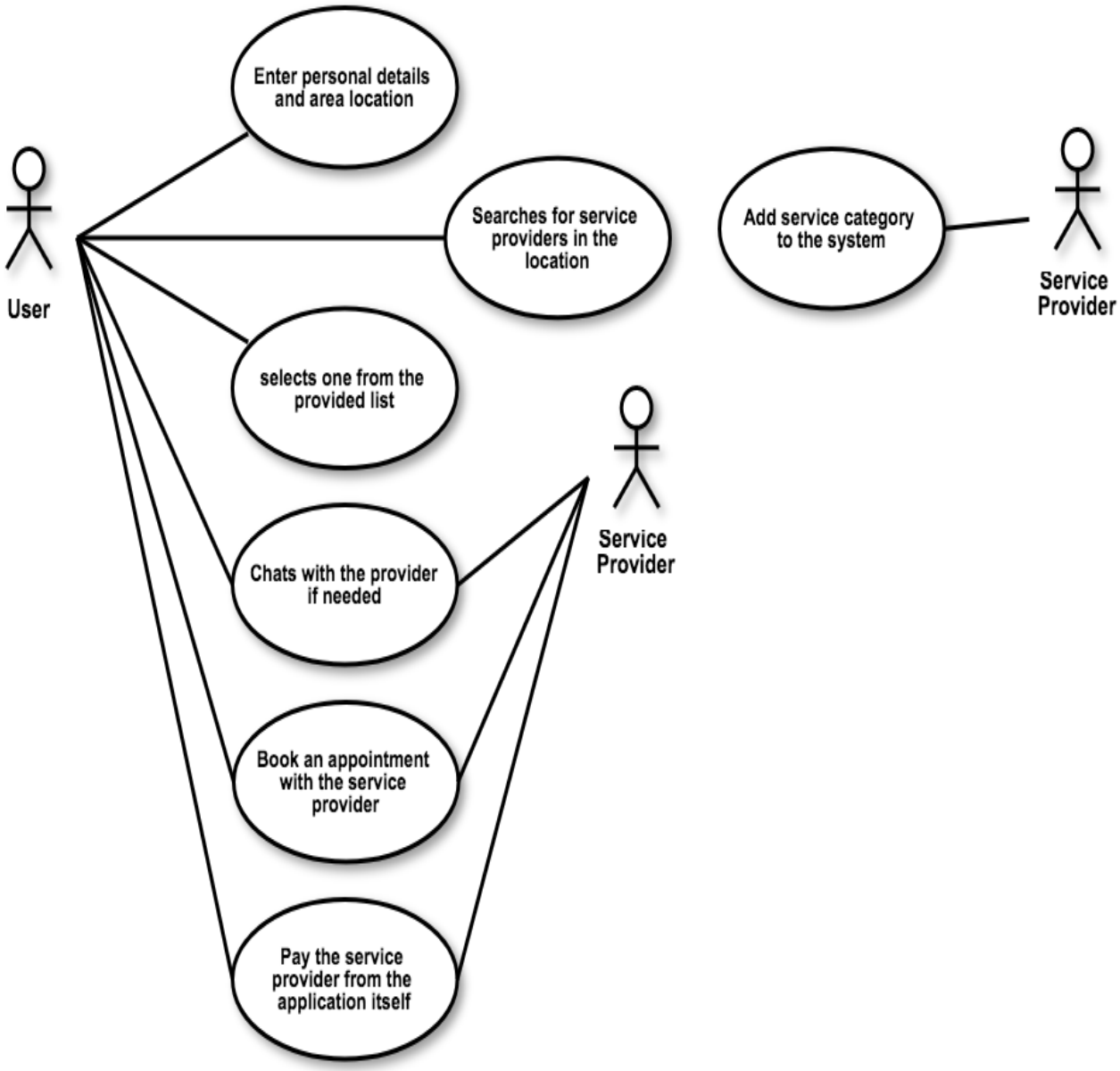
Chapter 2 discuss various aspects of the system such as requirements, design, architecture, implementation details, and testing of the application. Chapter 3 gives concluding remarks and Chapter 4 explores the further development opportunities in the application.

2 System

2.1 Requirements

This section list downs functional and non-functional requirement for EazyServices product.

2.1.1 Use Case Diagram



2.1.2 Functional requirements

Location Based Search Service

Users will find service providers based on their location. When the user logs into the application, its location will get updated and based on the location a listing of service providers will be presented to the user and the filtering criteria on this listing will be user location. This is the most important requirement as it forms the main matchmaking engine of the application.

Reviews and Ratings Service

Listing of service provider will be sorted based on favourable reviews and ratings received by them using the semantic web technologies. Priority will be given to service providers who have received rating or review from user's friends on social platform such as Facebook. This requirement forms the basis of search results being presented in the earlier requirement.

Appointment Booking Service

User selects a service provider and is presented with a screen where the user will get service provider's contact details such as phone and email. Along with this user will get an option to make an appointment with the provider or provide a review/rating. This service is the actual delivery point of the whole application.

Robotic Process Automation

Users will have an option to chat to a bot and book a service provider for their service. By using this functionality, the user can tell the

service provider the exact description of the service he/she expects. They can also agree to a price if needed. The functionality is not that important and has been added to make the application a more rounded product.

Payment Service

User will have an option to pay a service provider for services they have received through the application itself using a payment gateway. The payment gateway will be implemented using a stub rather than an actual implementation as it involves licensing cost. This requirement is important as it will provide monetary benefit to both the service provider and the development team.

2.1.3 Non-Functional requirements

Logging

Logging is important tool to monitor execution of code and use it as debug tool to understand how and what went wrong. EazyServices' middle tier will use Java Log4j logging capability to log sequence of control flow.

Authentication

Authentication is mechanism which provides only intended access to application. EazyServices will use multiple authentication capabilities to verify user's and administrator's credibility. A Spring based security capability will be integrated with system as default authentication mechanism. In most of the cases a system administrator will use this form of authentication while managing application. For social platform

based reviews and rating we will allow user to be authenticated using social media platform such as Facebook.

Authorization

Authorization gives authenticated user and administrator to see and act based on their granted permissions. EazyServices' database will have multiple roles and capabilities. Each capability will be preconfigured with a set of actions that the application user can perform. A role such as application user or application administrator will have a set of preconfigured capabilities. A user or administrator will be a part of single of multiple roles. When user or administrator login to application his or her role will be determined, and based on role appropriate actions will be enabled or disabled.

Scalability

Scalability is a capability of application to grow with load. It can be tested by adding more requests or processing load and can be achieved by adding more hardware capability and balancing (sometimes, distributed) load on various hardware devices. The EazyServices application will be scalable with front end web server distributing the request load on various application server instances. Each instance of application server will have its own independent database which will have replicated copy of data. The data between the database instances will be sync periodically. The application will be tested with vertical scaling technique.

Availability

EazyServices will be available for most of the time by deploying it and making it accessible over multiple instances. So, availability will be achieved using horizontal scaling.

2.1.4 Interface requirements

The following section describes responsibility, provided APIs and expected APIs for each component of the application EazyServices.

Registration System

This component provides the actor the functionality to register as a user/service provider/administrator with the system. The option to register will be provided when the user signs-up with the application for the first time.

Provided APIs

Register: This API will be used to register actor as user/service provider/administrator.

AddUser: This API will add user into the system database along with their credentials and the role that they have selected.

Expected APIs

None

Authentication and Authorization System

This component validates the user supplied credentials with the ones stored in the system database for the same user thus providing authentication services on log-in. It also provides the authorization services by checking user role at the time of login and providing the appropriate capabilities based on user role.

Provided APIs

Authenticate: This API will authenticate the user supplied credentials against system stored credential to validate use identity.

Expected APIs

None

User Management System

This component provides user management services based on user role. It provides user details and servers as the central spot for users from where they can access other services provided by the application.

Provided APIs

None

Expected APIs

AddUser from Registration System

Authenticate from Authentication and Authorization System

Notify from Notification System

Pay from Payment System

Search and Listing System

This component will provide users with the functionality to search service providers based on the service requirement and desired location. The user will get a list of service providers registered with the application based on the service and the region they provide on the screen.

Provided APIs

Search: This API will allow users to search service providers based on the service and the location they provide on the screen.

Expected APIs

GetReviewsAndRatings from Reviews and Rating System

Appointment Booking System

This component will allow users to book appointment with the service provider of their liking at the time and location they choose. It also provides the option to supply a name and the description of the appointment they are making.

Provided APIs

Book Appointment: This API will allow users to book appointment with the service providers they choose from the search listing.

Expected APIs

ConfirmAppointment

Reviews and Rating System

This component will allow users to provide reviews and rating to the service providers from whom the user has already availed the services. The rating will be out of 5 and the review will be in text format and will be optional.

Provided APIs

ProvideReviewAndRating: This API will allow users to provide review and rating to the service providers who have provided their services.

GetReviewsAndRating: This API will provide the functionality to get all review and the average rating received by a service provider upon request.

Expected APIs

None

Payment System

This component will allow users to pay the service providers upon successful service completion from the application itself. A proxy gateway has been implemented as an actual payment gateway incurs licensing costs.

Provided APIs

Pay: This API will allow users to pay the service providers for their services.

Expected APIs

None

Notification System

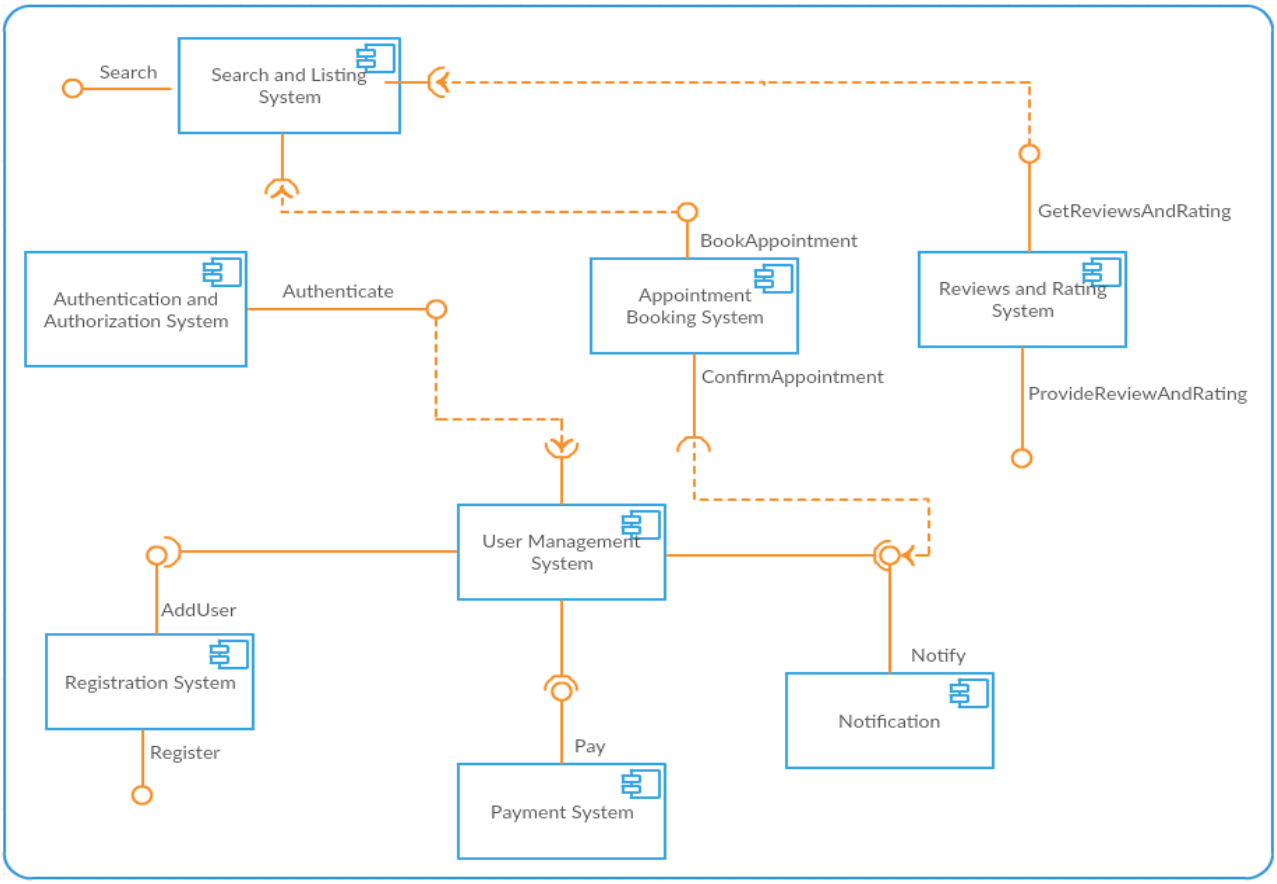
This component will provide e-mail notification to the users on appointment confirmation.

Provided APIs

Notify: Used to send e-mail notification to the user upon appointment confirmation.

Expected APIs

None



2.2 Design and Architecture

This section describes the technical architecture of EazyServices solution. The architecture has three tiers; client, mid-tier and data tier. All tiers are modularised and have no direct dependency on each other. The modularisation helps in development of each tier independently so that it can evaluate and vary without affecting another tier.

Client Tier

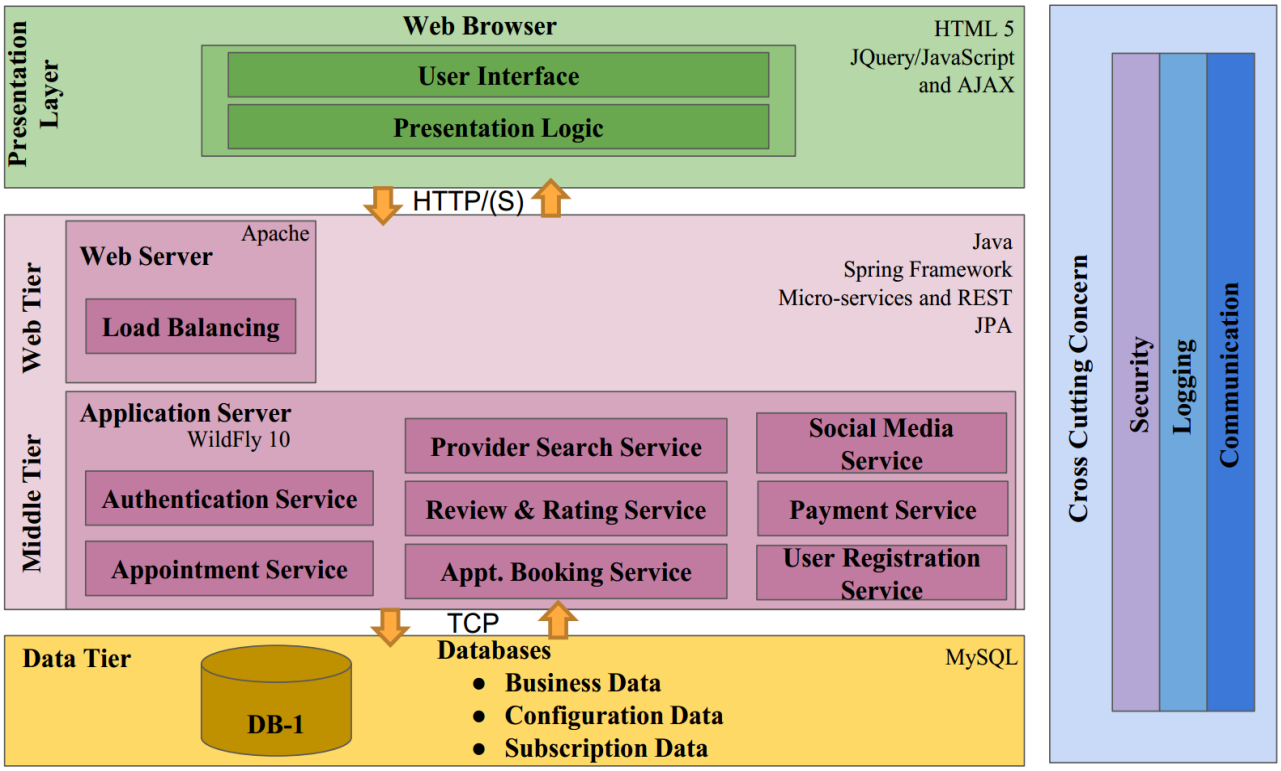
The EazyServices solution is available to users as a traditional browser based thin client. The client works on almost all popular

browsers such as Google Chrome, Mozilla Firefox and Microsoft Internet Explorer. The client is developed using HTML 5, JQuery, Java Script and asynchronous AJAX technologies. The client is RESTFul and uses Representational State Transfer like calling mechanism to communicate with mid-tier. It uses HTTP as base protocol for communication however if the web server is configured to work on Secured Socket Layer then it uses HTTPS protocol for communication.

The presentation tier is mainly divided into two layers; user interface and presentation logic. The user interface layer is responsible for showing Graphical User Interface to user whereas the presentation logic layer decides when and what to show to user based on interaction. The presentation layer is developed using HTML 5 and Cascaded Style Sheet (CSS) whereas the presentation logic layer is developed using JQuery, Java Script and AJAX technologies. The following figure shows the presentation tier with layers and technologies used.

Middle Tier

The mid-tier of EazyServices consists of two part, the web server and application servers. The web server is front server which trap the user request and then redirect the request to application server. At the response time the application server first sends response to web server and then web server sends response back to requesting client.



The Web Server

Apache web server is used as front end server to handle all user requests. The web server is configured to run on port 80 and works as manager. Multiple application server worker threads can be created to serve user request. In current implementation two application server workers are created. The default round robin fashion of redirecting user request to application server is used. The presentation layer contains only HTML pages and Java Script; therefore, these can be deployed on web server instead of application server for better performance. With this implementation, for static pages, web server can redirect response without reaching out application server.

The Application Server

Redhat Wildfly server is used as an application server for deploying services. However, as development of micro-services is done easily with Spring Boot framework which provides default Tomcat container, we adopted both (Wildfly and Tomcat) for micro-services deployment. The Java Standard Edition, Spring framework, micro-services development using REST and Java Persistence API (JPA) are used to develop mid-tier components.

Instead of using traditional monolithic web application approach EazyServices is developed using light weight micro-services using REST. Micro services are scalable, heterogeneous, provides clear contextual boundaries and ease of independent development and deployment. It is relatively easy to distribute and fault tolerant due to its self-contained nature and concise specification. Moreover, it is cloud ready and goes well with DevOps.

Data Tier

MySQL 5 is used to store business, configuration and subscription data in relational fashion. Database is only accessible by services defined in mid-tier. To access these services a user need to be authenticated with the system. Therefore, access to database is inherently secured from malicious users. Along with that, database is only accessible to authorized administrator from database clients or console.

Cross cutting concern

The cross cutting concerns such as logging, authentication, communication, management and configuration have been taken care across all tiers of application.

Logging

For logging at mid-tier application uses Simple Logging Façade for Java (SLF4J) interface to use logging implementation provided by Spring Boot framework. The logging level such as DEBUG, INFO, and TRACE can be configured using Spring Boot configuration properties file.

Authentication

For authentication, Spring security and Java Enterprise Edition security mechanism has been evaluated however that turnout to be configuration overhead for small application therefore finally a *Login Service* is written on mid-tier to handle user authentication.

Communication

The application uses HTTP and if web server is configured on SSL then HTTPS is the protocol used to communicate between presentation tier to mid-tier. MySQL database supports multiple communication protocols however to communicate between mid-tier to data-tier TCP is used.

System Management and Configuration

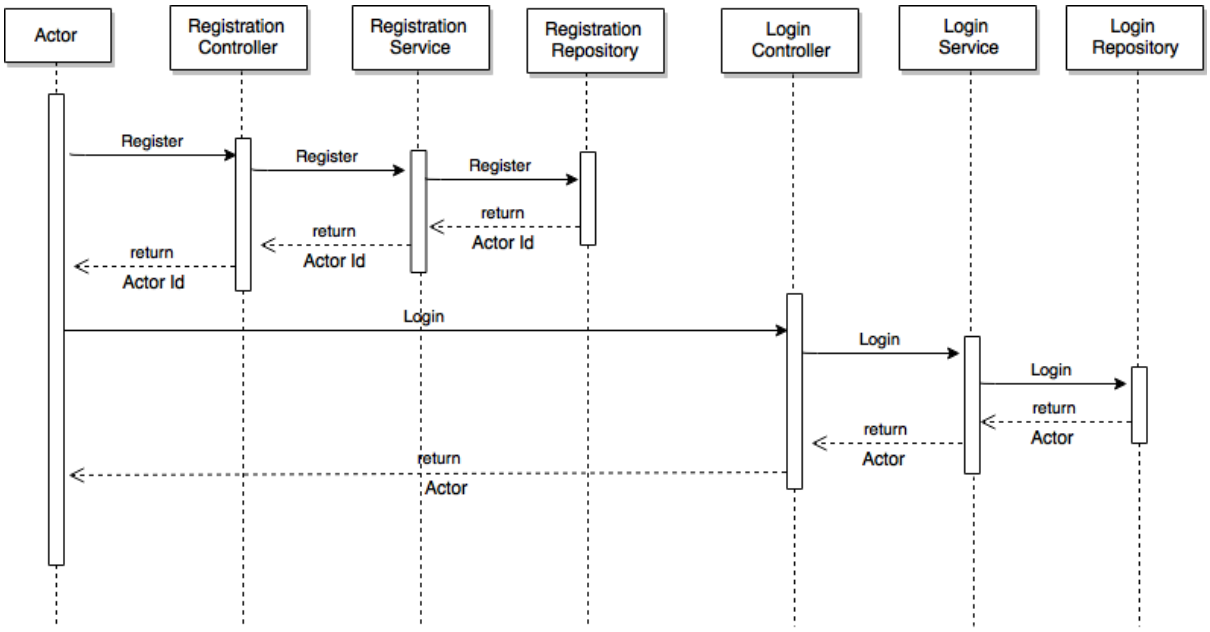
Failure of services such as web server, application server and database will be logged to configured log file. When the product is unresponsive then log file can be used to monitor activities and find out fault. Apart from that, these services have default administrator

console which can be used for management. Similarly, for configuration of product application server console can be used. Mostly, the application server needs lot of configuration which are handled by configuration properties file defined as part of source code. In most cases, restarting servers should solve the failure. In rest of the cases log file can be used to find out failures in the system.

The system can handle all types of crash failure of application servers because system uses multiple application servers on mid-tier.

2.3 Implementation

Registration-Login Service Sequence Diagram

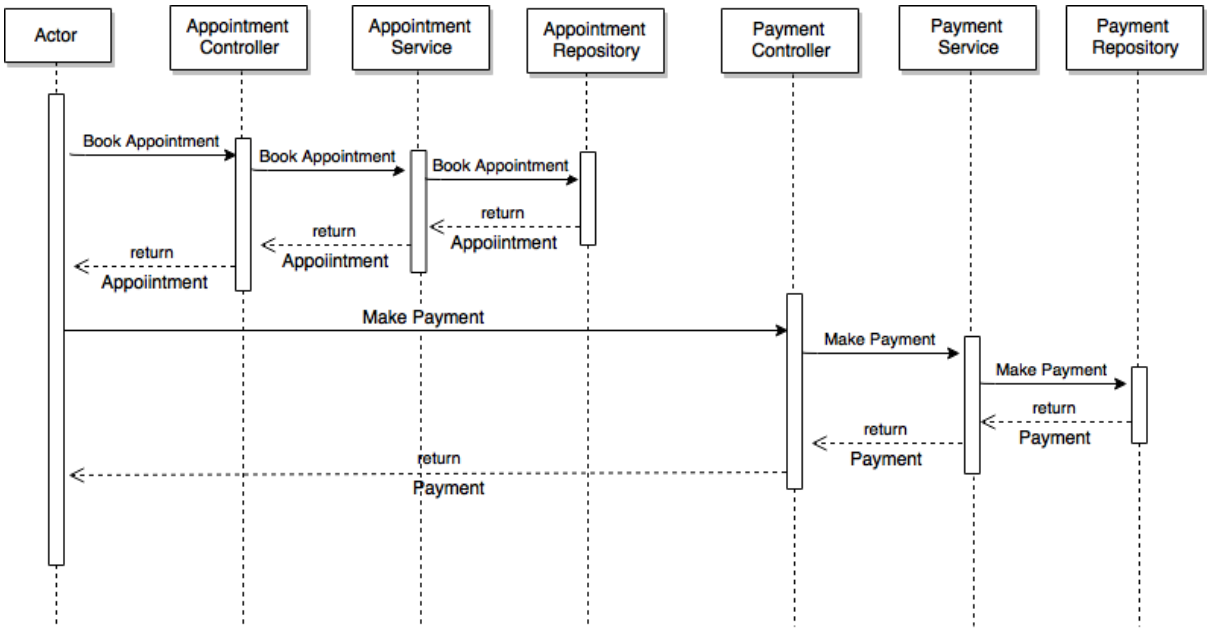


The above sequence diagram shows the sequence of interactions between the components while registering with the application and

then signing-in. The actor while registering, selects the role with which he wants to register. Currently our application supports 3 roles: *User*, *Service Provider* and *Administrator*. Post this the actor fills the corresponding registration form and sends the registration request to the backend. At the backend, *Registration Controller* intercepts the request, containing the registration details and the role type for the actor, and forwards it to the *Registration Service*. The *Registration Service* consists of the business logic to register the actor. The service first encrypts the password using SHA-256 and then based on the role of the actor, sends the registration request to the appropriate *Registration Repository* (User / Service Provider / Administrator Repository) which then takes care of the operations for storing the registration details in the database. Post registration, the *Registration Service* receives the actor id for the actor which it sends to the user interface via *Registration Controller*.

For login operation, the actor provides his/her credentials (username and password). The request for login is intercepted by the *Login Controller* which forwards the request to the *Login Service*. The service first contacts the *Login Repository* to fetch the credentials of the actor who wants to login. If the credentials fetched do not match with the ones provided by the actor, null object is returned and the message “credentials do not match” is logged. If the credentials match, the repository is again contacted to fetch the Actor object which is returned to the user Interface via *Login Controller*.

Appointment-Payment Service Sequence Diagram

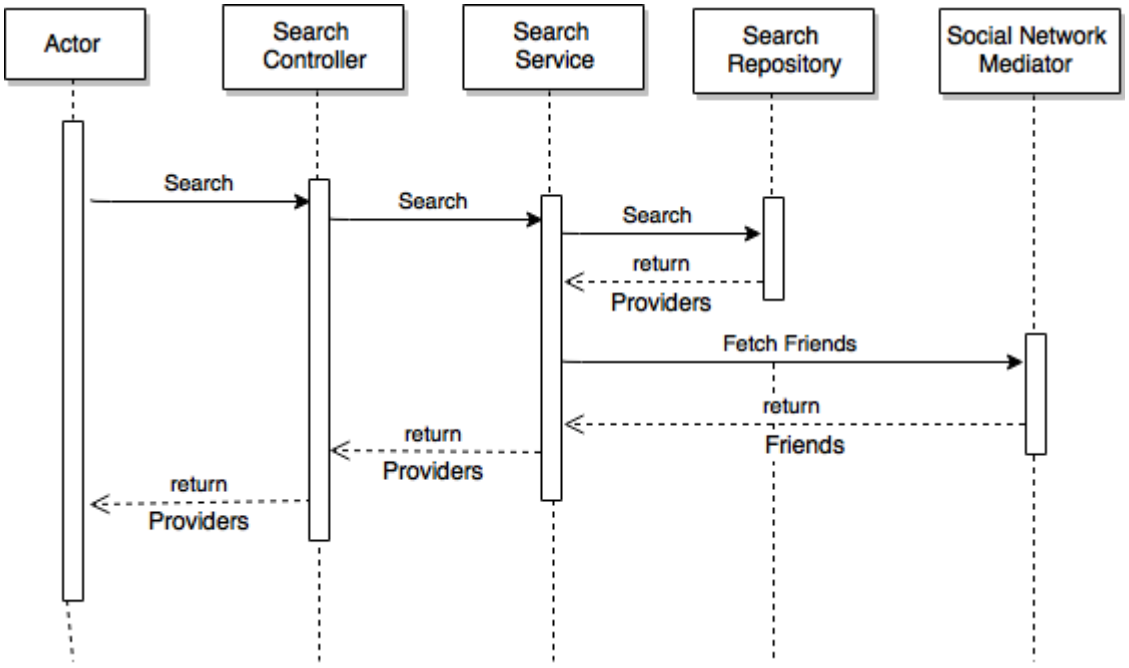


The above sequence diagram shows the sequence of interactions between the components while booking appointment and making its payment. The *Actor*, through user interface, sends the appointment booking request to the backend. The request, containing the *Appointment* object, is intercepted by the *Appointment Controller* which forwards the request to the *Appointment Service*. The *Appointment Service* consists of the business logic for appointment booking. The service sends the request to the *Appointment Repository* which takes care of all the DAO operations for *Appointment* bean. After storing the *Appointment* into the database, the service receives the *Appointment* object which is then passed to the user interface via *Appointment Controller*.

Post appointment booking the user can make payment using the same user interface. The request for payment, containing the

Payment object with the appointment id, is intercepted by the *Payment Controller* which forwards the request to the *Payment Service*. The *Payment Service* consists of the business logic for making payment of the booked appointment. The service sends the request to the *Payment Repository* which takes care of all the DAO operations for *Payment* bean. Since no third-party gateway is used, we have implemented a proxy gateway which does the payment transaction without checking any details. After the payment is done, the *Payment Service* receives the Payment object with status of transaction and sends this object to the user interface via *Payment Controller*.

Search Service Sequence Diagram



The above sequence diagram shows the sequence of interactions between the components when user searches for service providers

for a service. The user first provides the type of service he wants and the region in which he wants the service. The request, containing the *User* object with service type and region, is sent to the backend. At backend, the request is intercepted by the *Search controller* which forwards the request to the *Search Service*. The *Search Service* consists of the business logic for finding all the service providers with their reviews based on the requested service type and region identifier. The service contacts the *Search Repository* to fetch all the required list of service providers and their *Reviews*. Post this, the *Search Service*, using Facebook id of user, fetches all his/her friends on Facebook which are also registered with our application. For the previously fetched list of service providers, a sub list is prepared which contains the service providers who have been reviewed by the registered friends of the user. Both the parent list and sub list with their corresponding reviews are returned to the user interface via *Search Controller*.

Rating-Review Component Interaction

For rating the service provider post the service is provided, the user rates the service provider out of 5 and provides an optional review for his/her experience. The request containing the *User* and *Service Provider* objects with the rating and review is sent to the backend. At backend, the request is intercepted by the *Rating controller* which forwards the request to the *Rating Service*. The *Rating Service* consists of the business logic for storing the rating and review for the service provide. The service sends the request to the *Rating Repository* which takes care of all the DAO operations for *Rating-*

Review bean. After storing the rating and review into the database, the service receives the *Rating-Review* object which is then passed to the user interface via *Rating Controller*.

System Structural Model

The structural model of EazyServices gives a view of a system that emphasizes the structure of the objects, including their classifiers, relationships and attributes. The operations structure is intentionally skipped to save space as it can be inferred from the class name.

Model-View-Controller architectural pattern

At very high level, figure 3 shows the Model-View-Controller architectural pattern used in product.

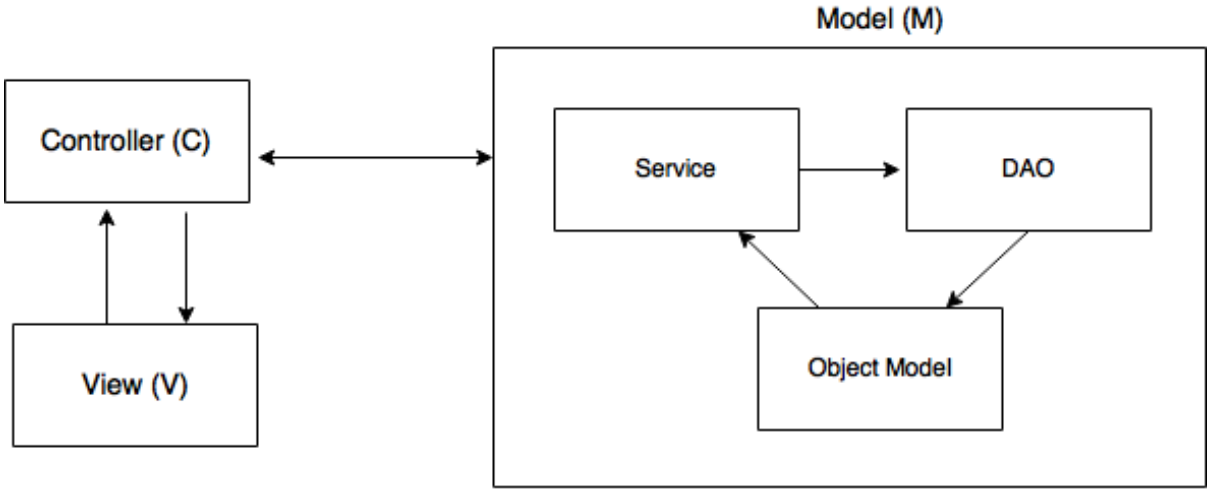


Figure 3. Shows the Model-View-Controller architectural pattern used in product

The *view* consists of multiple HTML pages rendered in browser. The *controllers* are *Plain Old Java Objects* (POJO) and written in Java using *Spring Boot* framework. The model comprised of *services*, *Database Access Objects* (DAO) and *object model*. The *object model* is described in following section. Like *controllers*, the *services* and *DAOs* are POJOs and written using Spring Boot framework. The *views* are deployed in web server and the *controllers* and *models* are deployed in application server.

Class Diagram

The figure 4 shows static modelling of classes, their associations and attributes. At the heart, it has actor and each actor is associated with *Role*. There are three types of actors involved in system, namely *User, ServiceProvider* and *Administrator*. There is no association between actors, however, each actor has a *Role* to play. An actor cannot play more than one *Role* therefore it has cardinality one-to-one. The *Role* class defines which role an actor can play. There are three types of roles; user, administrator and service provider.

A *Credentials* class is independent and treated as *value object*. In real world, credentials are associated with actors, however, in static modelling purposely the association is missing because of security concern.

A *Service* class represents type of service offered by service provider. A service provider can provide multiple services such as

carpentry, house cleaning and electrician etc. Similarly, a service can be provided by many service provider. Therefore, cardinality between service provider and service is many-to-many.

An *Appointment* class depicts the real-world job. When a user has some service to perform then the user first book an appointment with service provider. The appointment has information about user who is booking appointment, service provider who is going to serve, appointment date and service to provide. User can book just one service per appointment therefore user to appointment cardinality is one-to-one. Similarly, only one service can be serve per appointment therefore cardinality between service and appointment is one-to-one.
A *Payment* class calculates actual amount to pay against the service offered by service provider. Once the specified work is completed by service provider then user is liable to pay service charges. User can look for completed appointments and pay using this proxy payment gateway. The cardinality between payment and appointment is one-to-one.

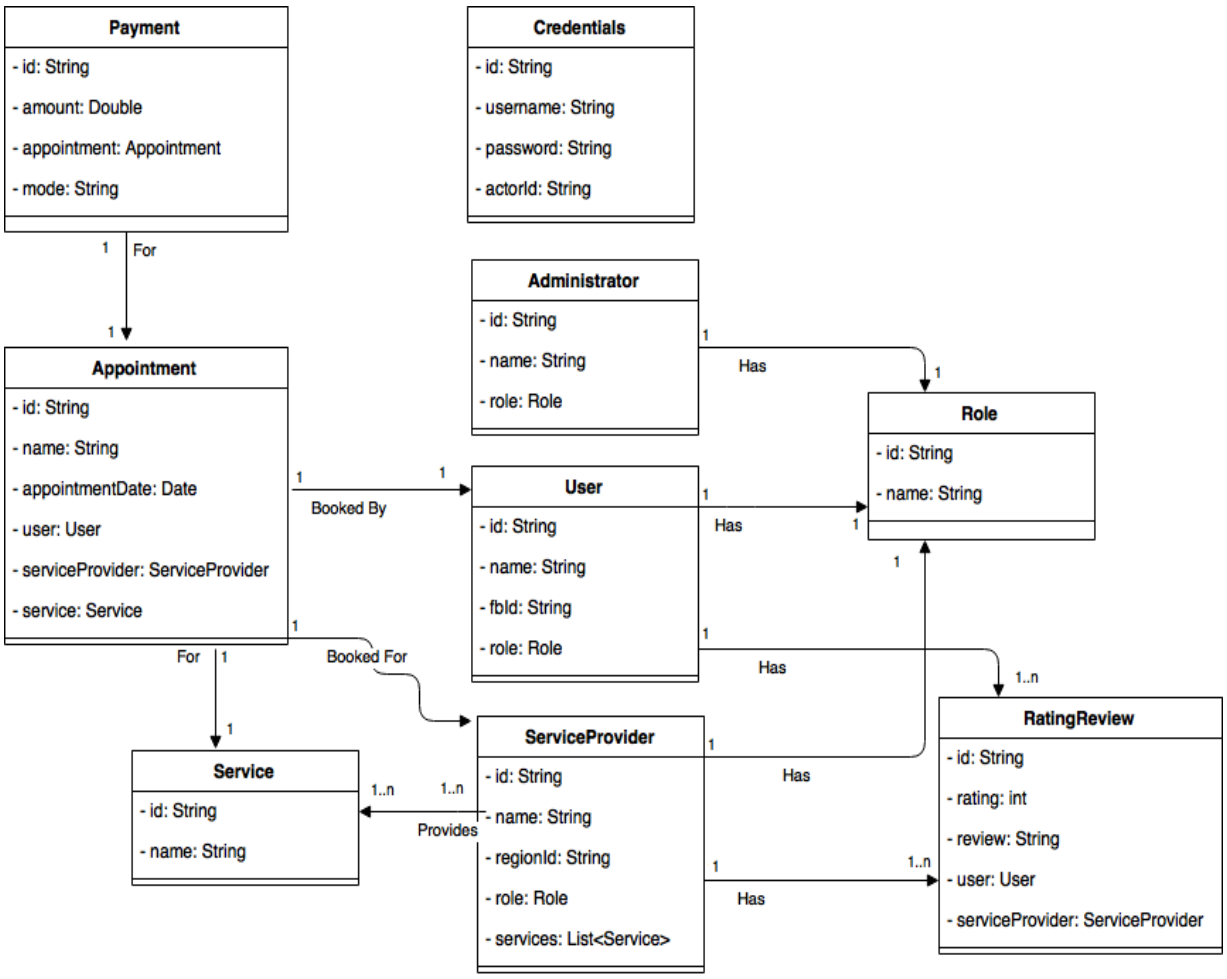


Figure 4. System structural model of EazyServices product (Class Diagram)

A *RatingReview* class does two functions, let user rate the service provided by service provider on scale of five (five being highest) and also let user write review. As a user can write multiple reviews per appointment therefore cardinality between user and review is one to many and similarly a service provider can have multiple reviews therefore cardinality between service provider and review is one-to-many.

Entity Relation Diagram

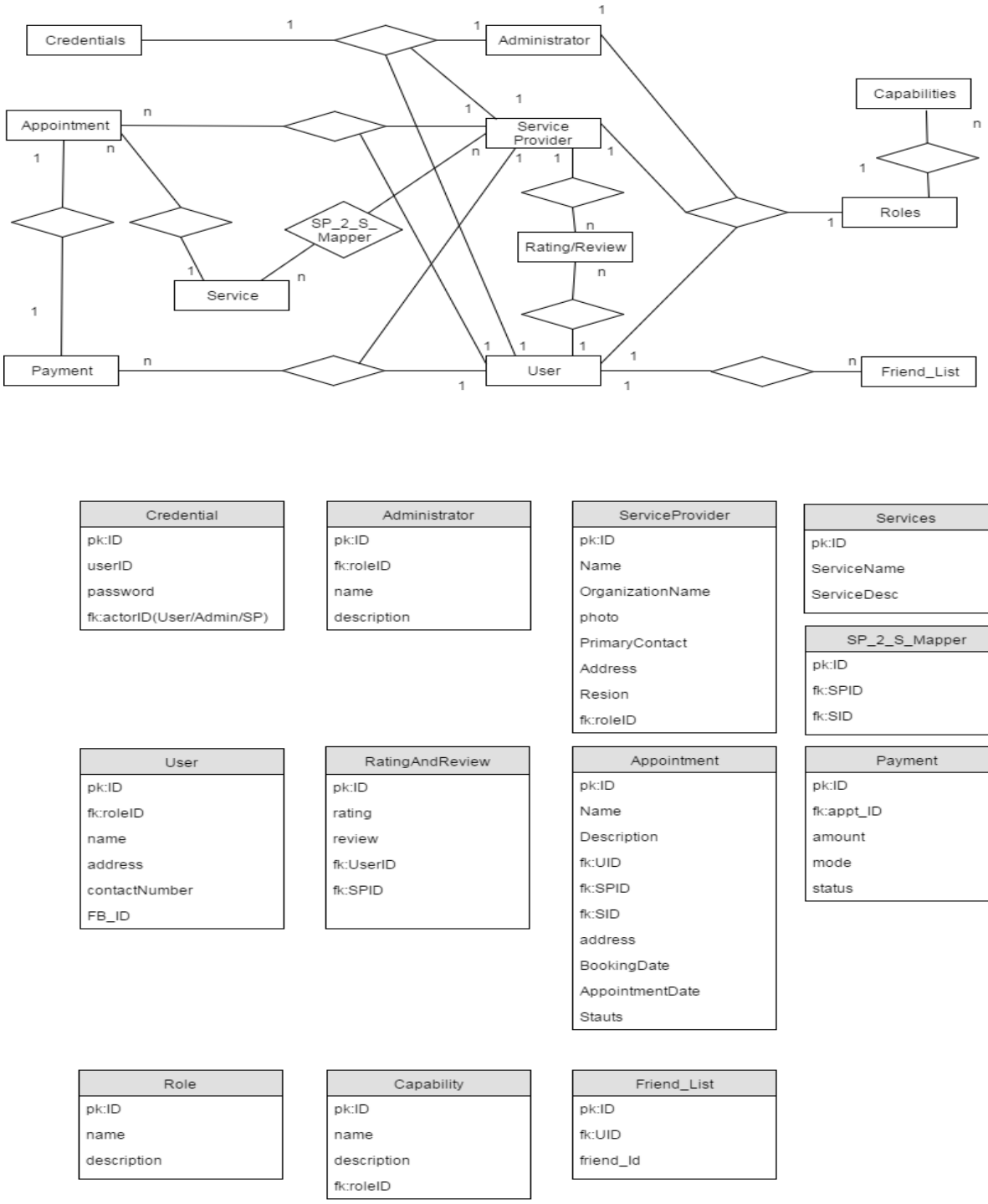
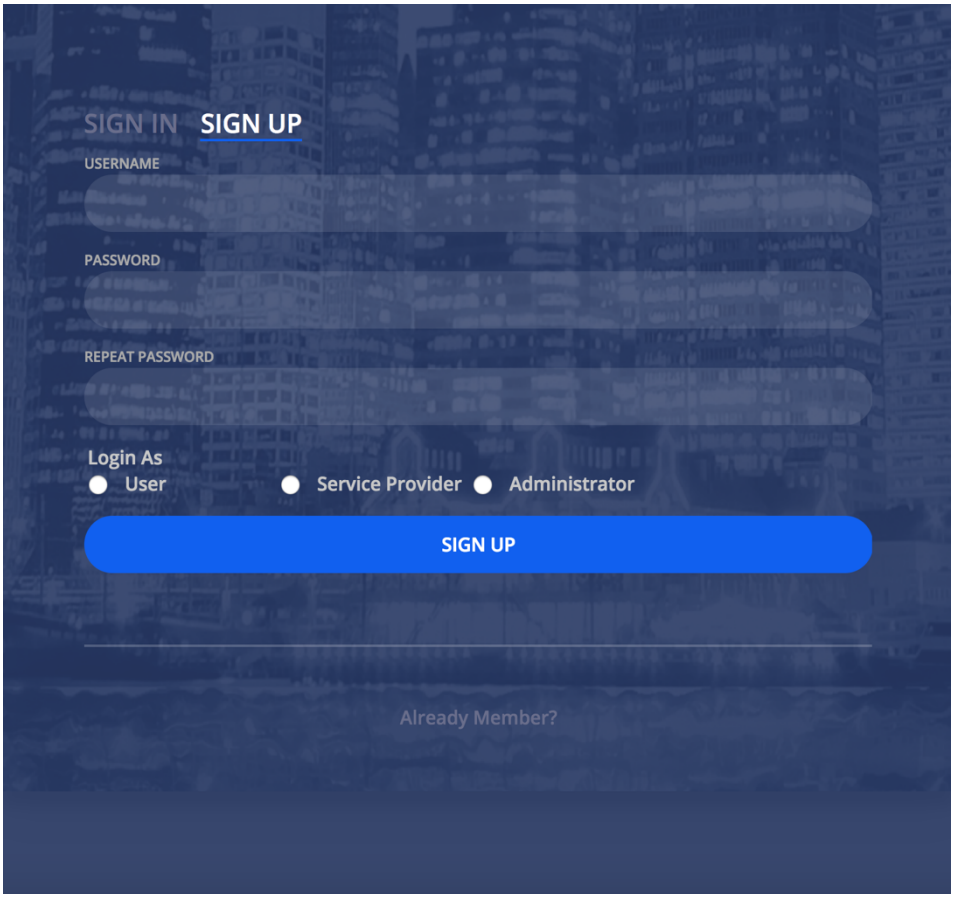


Figure 5. System structural model of EazyServices product (ER Diagram)

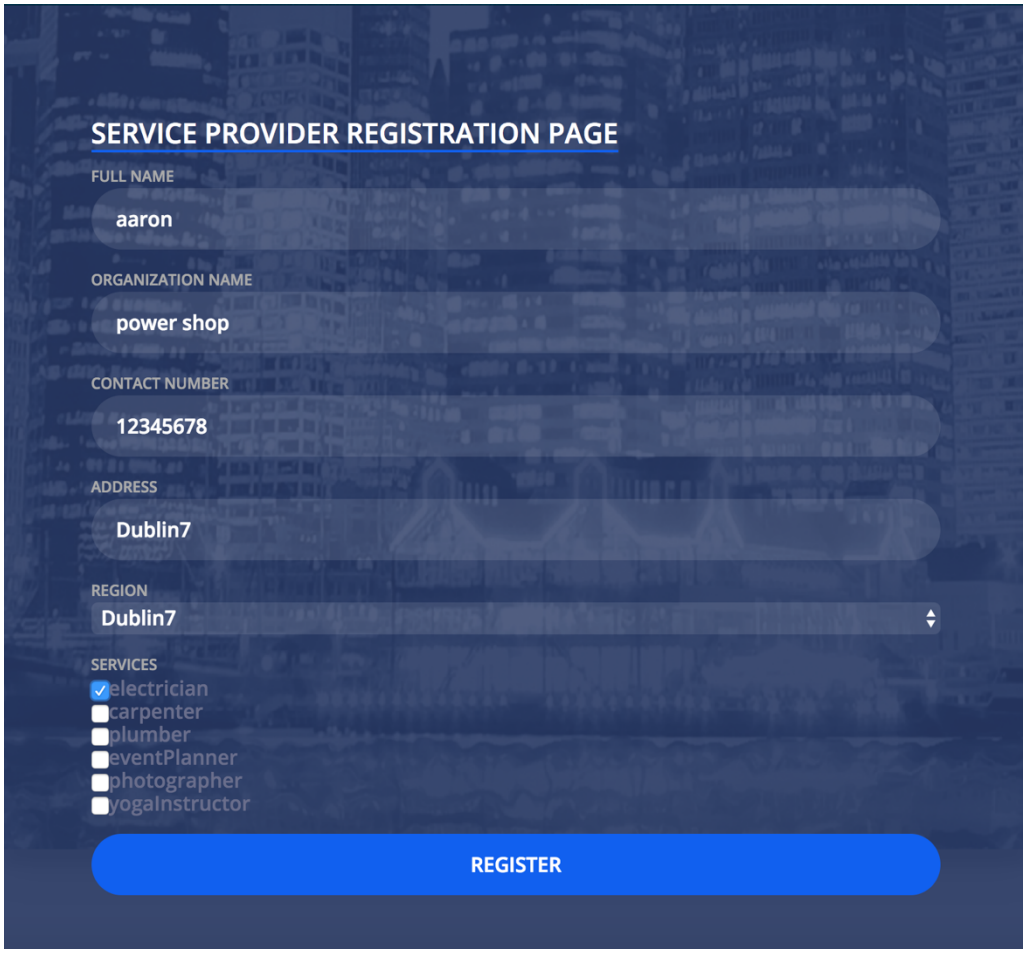
Generally, the entity relation and object model are different from each other, however, the EazyServices being a small application we manage to keep both same so that there is one is to one mapping between the two models. Moreover, the ER model has couple of more tables than the object model and those are called a mapper. The mappers are basically a table which holds many-to-many relationship between objects such as service and service provider.

2.4 Graphical User Interface Layout

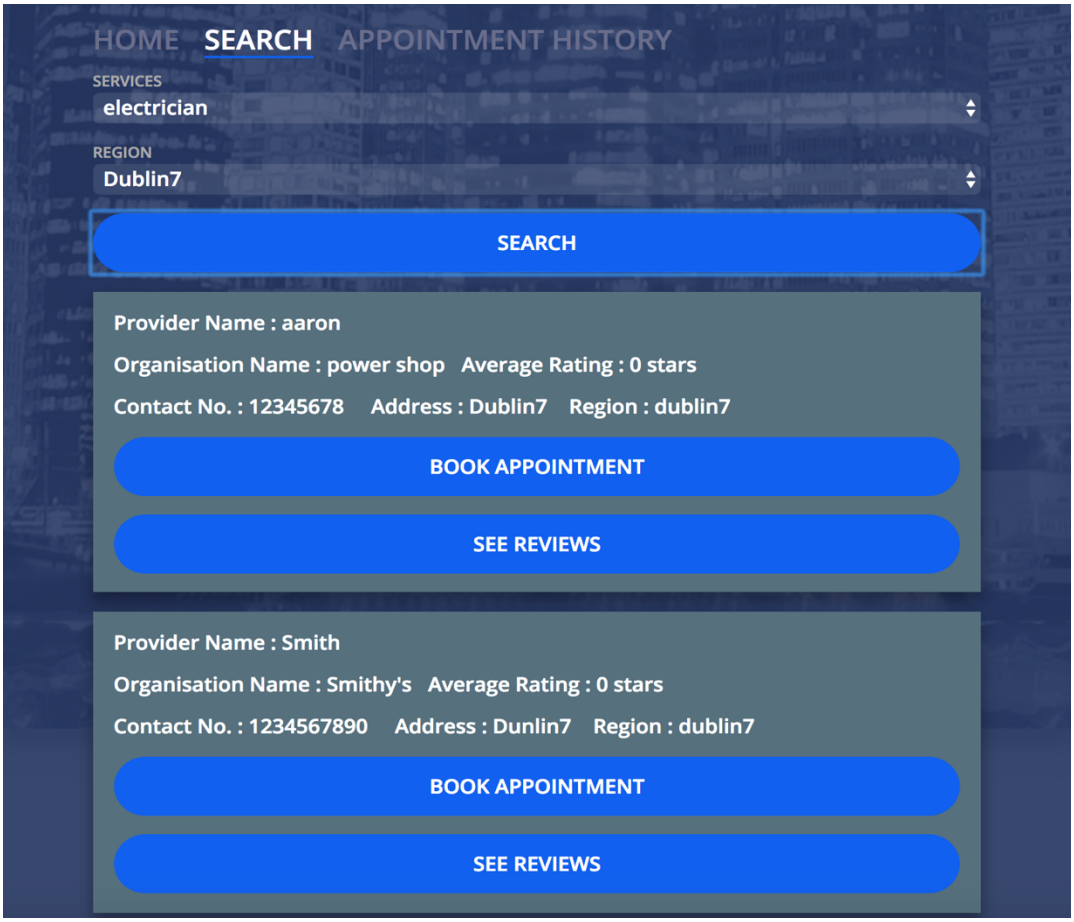
This section provides screenshots of some key screens in the application and explains each one of them.



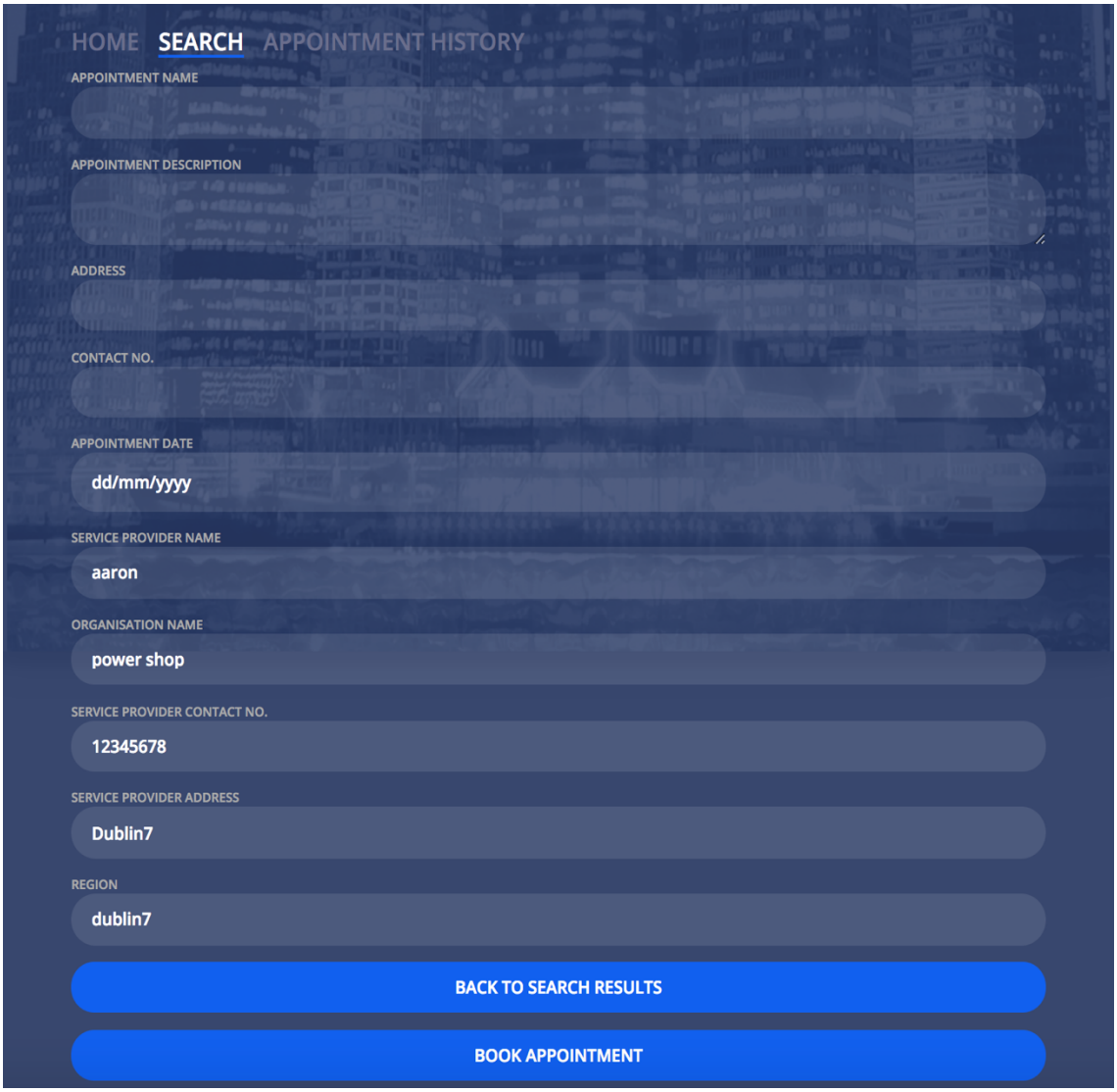
This is the sign-up scree where user can sign-up as a service provider or as a service user.



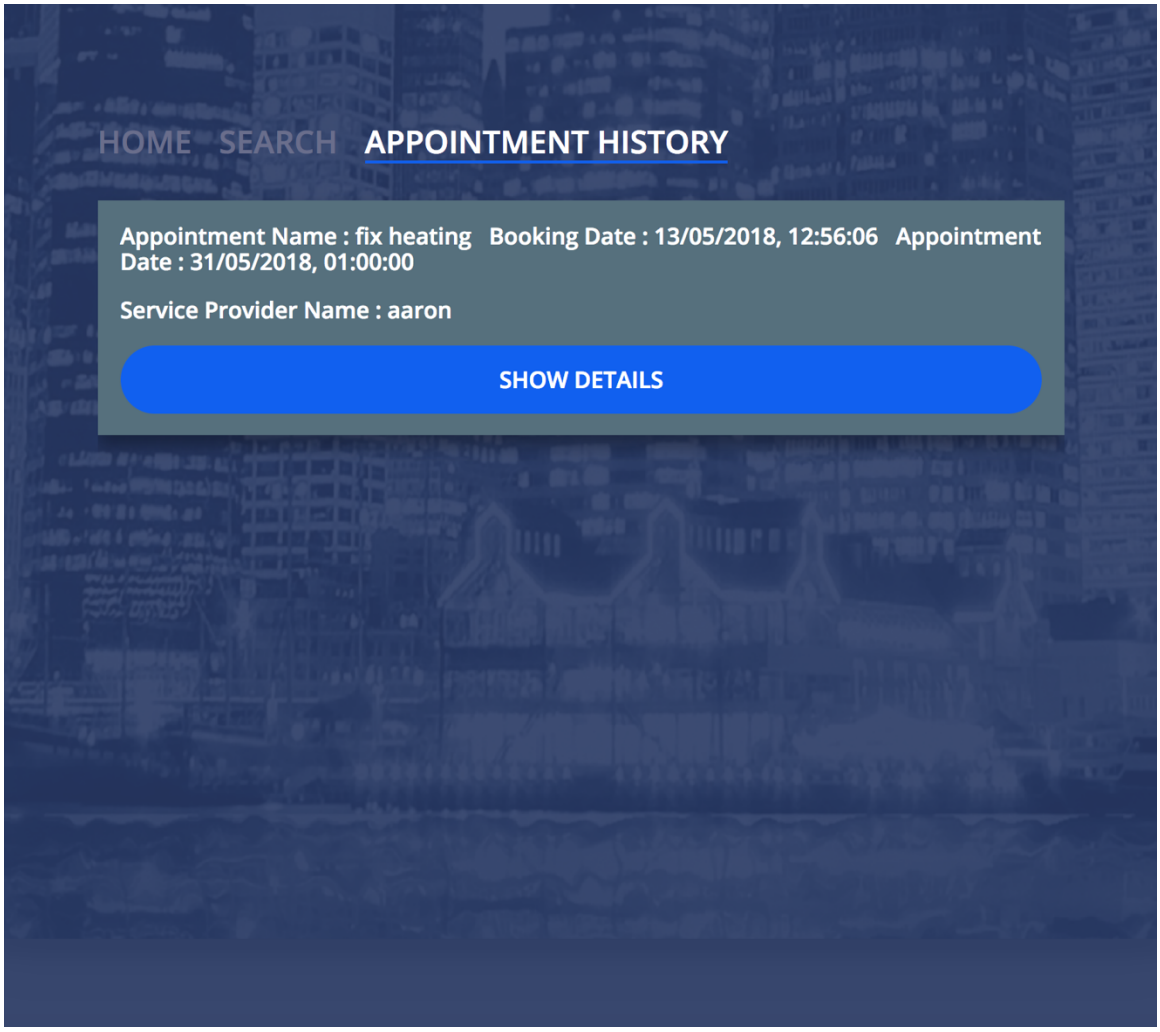
This screen shows registration process for a service provider. Apart from contact details a provider is supposed to give their region for operation and their services. A service provider can provide multiple services if they wish to do so.



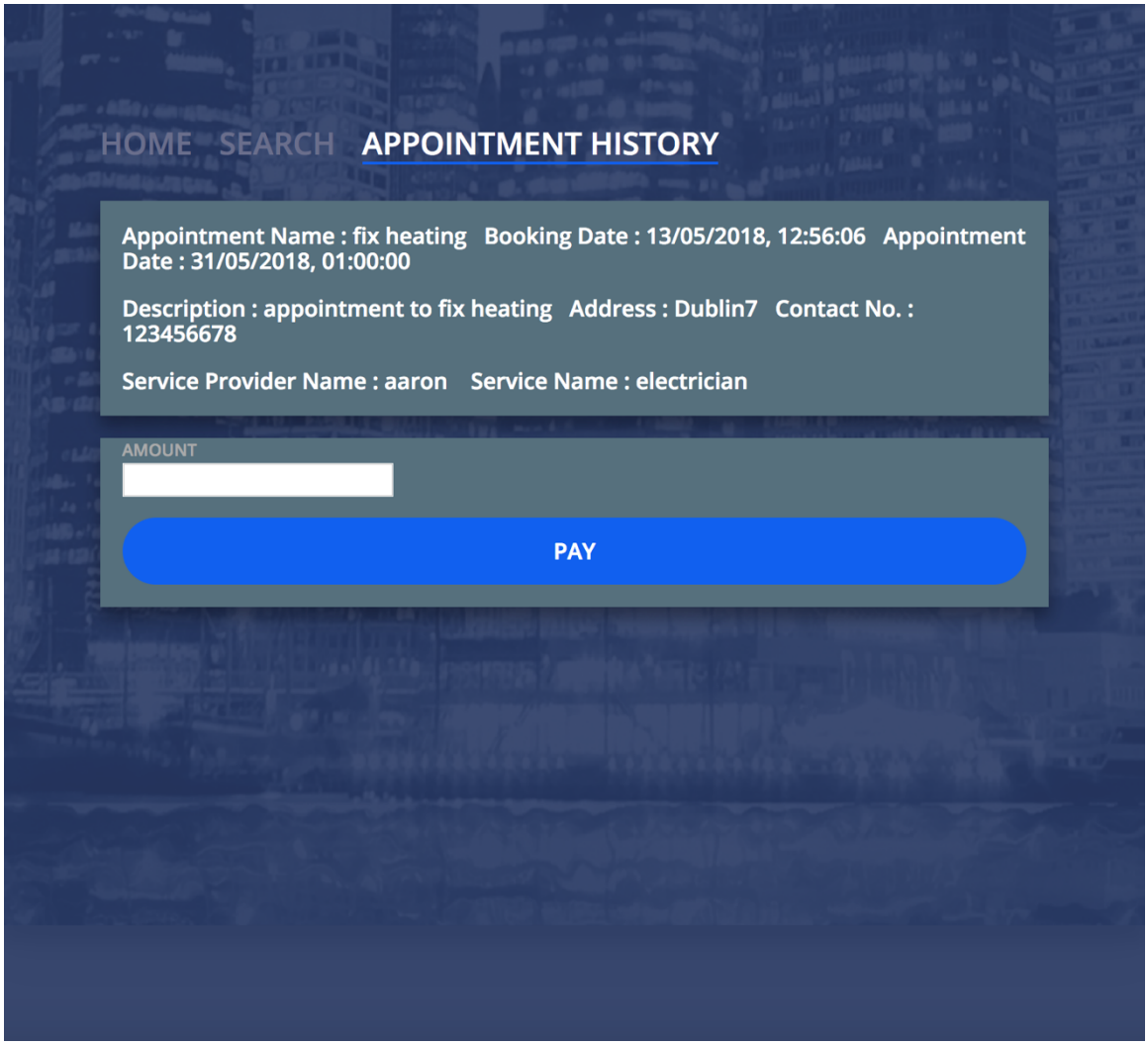
This screen shows search functionality in action. It lists the providers based on previous ratings calculated from their reviews. Here, since no review had been done so the average rating is 0. The user can write a review or book the services of the provider.



This screen shows the appointment booking process. The details of provider are prefilled and the user has to provide details like description and date for the appointment.



This screen shows all the appointments a user has made and can view them in details as well.



Clicking on show details also allows the user to pay for the service provider by using the PayPal sandbox.

2.5 Testing

Test Plan

A test plan is a document detailing the objectives, target market, internal beta team and the processes involved in a specific beta test for a software or hardware product. I developed a test plan to ensure that all the testing I wanted to carry out was achieved. The different types of testing methods I had planned to carried out can be

compared to the actual testing methods carried out below, the green highlighted fields display the testing methods that were performed . A copy of my test plan can be viewed below.

Approach	Type Of Testing	Manual Testing
Standard Testing	Integration Testing	Yes
	System Testing	
	Usability Testing	Yes
	Acceptance Testing	Yes
	Verification	Yes
Special Types of Testing to address specific challenges	Compatibility Testing	Yes
	GUI Testing	Yes

Integration Testing

Integration testing is the phase in which individual features or modules are combined and tested as a group this type of testing occurs before validation testing. Integration testing was performed throughout my project every time a new feature or functionality was added as it helped me see how all the functionalities connected and worked together. I had to particularly perform this type of testing after I added the login and registration functionality to ensure that there

was a communication with the database. When user sessions were added this type of testing helped ensure the functionality was working as a communication between the login and the user session was vital to store the user session and display the user's correct details and give access to the booking services. Results were recorded and any errors found were fixed.

Verification

Verification is the process to make sure that the application satisfies the conditions imposed at the start of the development phase, it is done to make sure the application behaves the way it is designed to and for the purpose its built. Verification has been conducted though out the application to make sure each feature behaves the way it should per the requirements. It is very important to conduct verification on all components of the application to avoid the release of a version with bugs. After conducting verification on all features, I discovered that the reviews system was not working correctly and so I fixed it on time.

System Testing

System testing is testing to ensure that the application works in different environments (older browsers, different systems) as expected. System testing is done with full system implementation and environment. This type of testing falls under the class of black box testing. I did this testing manually and used different versions of

popular browsers like Google Chrome, Mozilla Firefox, Microsoft Edge and Apple Safari.

Acceptance Testing

This type of testing is done by the customer to ensure that the delivered product meets the requirements as the customer expected. A test case helps with acceptance testing as a test case is a set of conditions or variables under which a tester will determine whether a system test satisfies requirements or works correctly.

A test case was developed by myself and my brother was asked to test the application as he had never seen my application until today. While he was testing the application, I was busy documenting all the results. I then took the results recorded and started to fill out the test case with the results I had written down. I began to document the step details my brother took, the expected results, the actual results and if the step passed or failed. After I documented the results I realized I had to make some changes and improvements to my application. This testing helped me realize the errors a customer could have faced if they were not repaired before app release. Some recommendations were made by the tester and they have been added.

GUI Testing

GUI testing is the process of testing the systems graphical user interface of the application under test. GUI testing involves checking

the screens with the controls like menus, buttons, icons and all types of bars etc. GUI is what the user sees and it is important this type of testing is carried out to ensure a fluid and error free GUI is offered to the user. I had performed GUI testing to make sure that all the buttons such as ‘Book Now’, ‘Search’ etc. we’re functioning correctly and the GUI was displayed once the button had been clicked. All other screens of the application were functioning correctly.

3 Conclusions

The goal I set at the start of the year was nearly fulfilled. I believe that I developed the application to the best of my abilities. The application development project gave me a good exposure to various technologies and state of the art technology frameworks like Spring boot, Facebook graph API, and Semantic web. I believe the skills I learned through this module would help me in my later career. I also tried to complete implementation for robotic process automation but was unable to do so due to lack of time.

The application development process was very enjoyable and the results of the application show the amount of time and effort that were applied into the development of this application. I hope to be developing this type of project in the future.

4 Further development or research

System could evolve over time by extending it to various platforms like mobile phones. An app can be made that would make the portal more accessible to wider audience. The app can also be extended by providing support for virtual wallet that could provide the facility of cash back which can be used for later services.

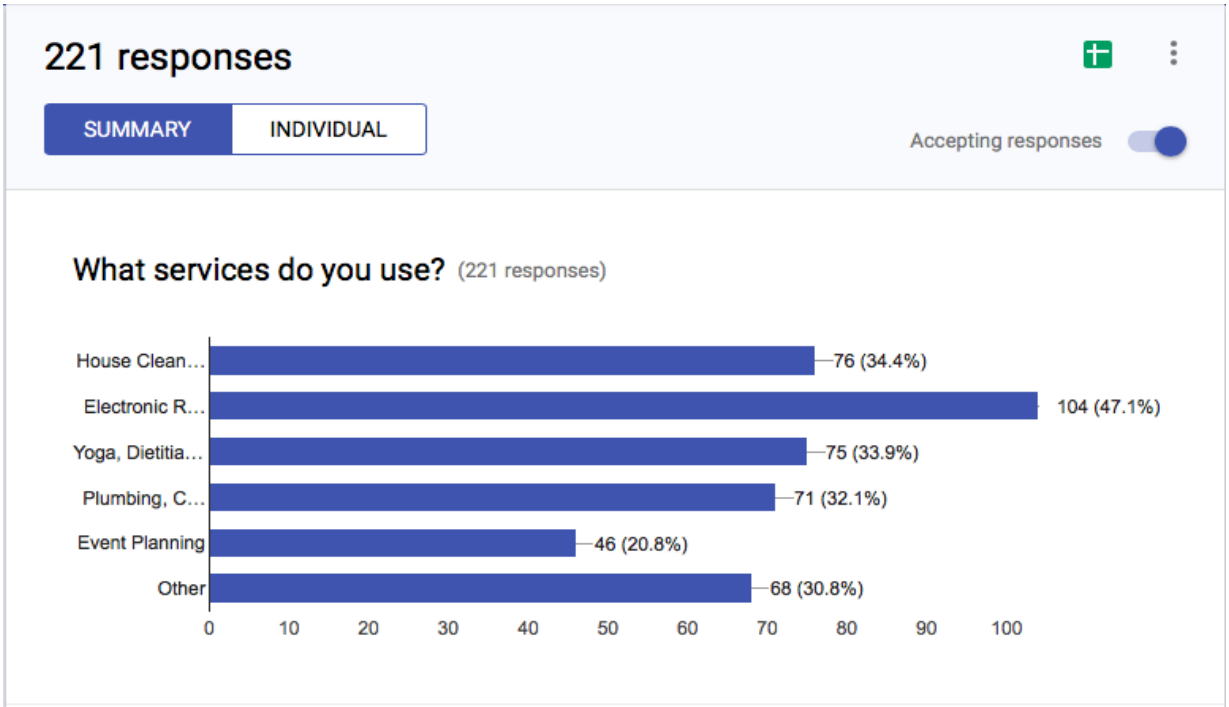
5 Appendix

Code Repository

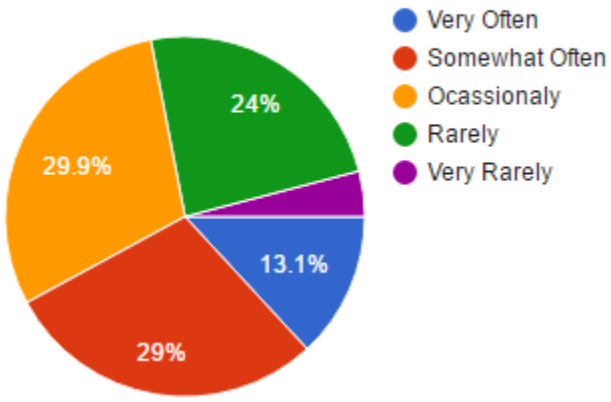
https://github.com/KhateebAhmad/EazyServices.git

https://github.com/KhateebAhmad/EazyServicesClient.git

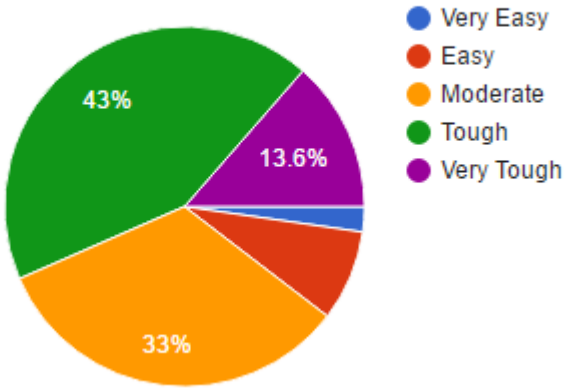
Survey questions

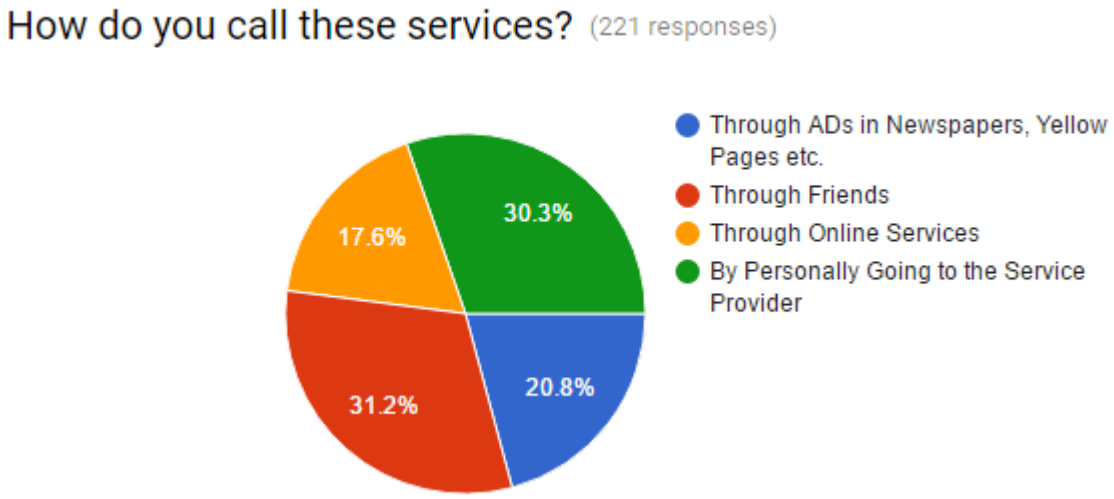


How often do you need these services? (221 responses)



How easy do you find to call these services? (221 responses)





Monthly Journals

Reflective Journal

Student name: Khateeb Ahmad

Programme: BSc. (Hons.) in Computing

Month: September 2017 – December 2017

My Achievements

In these months, I could finalize the following things:

Approach of project

From the beginning, I had decided to follow agile methodology for product development. I had to myself play different roles (such as developer, tester, product owner and client) over the life cycle of product development. At the start of project, I acted as product owner and client.

Product Conceptualisation

At the start of project, I listed down all function and non-functional requirements. I then did a story writing session where each functional requirement was broken down into a fine grained user story. During this period, I also prioritised user stories. Based on priority I created and maintained a product backlog that contain all user stories.

Sprint Planning

After that I estimated product efforts and relative size of development goals. I took each story one by one and assigned a story point based on development, testing and configuration complexity. Once each user story was assigned with story points I divided user stories evenly based on story point into sprints or iterations.

Designing and Prototyping

Before starting actual development, I made mock wireframes for the graphical user interfaces and how that will communicate with mid-tier services. I also made a high level conceptual design to define which component will be placed at which location. I also spent quite a good amount of time on designing object and data models for the application.

Development and Testing

I used test driven development model (TDD) which helped me to strictly adhere to design and goal of user story. It also means that I don’t have to spend lot of time at the end of project in fixing bugs.

My Reflection

I felt that by planning in advance it made things easier and there were no surprises later on. My development process also went a lot smoother as everything was pre-planned and I knew all dependencies early in the process and was thus able to plan a course for project completion.

However, I felt the process would have been much smoother and fruitful if I had some team mates and was not working alone. Since, I worked alone, I had to fulfil a lot of roles which meant that sometimes there was scope of things getting missed or a requirement not getting covered.

Intended Changes

Next on the agenda is to continue project development and to get ready the complete UI for the next month. I would also like to complete some basic functionalities as well.

Supervisor Meetings

Date of Meeting:

Items discussed:

Action Items:

Reflective Journal

Student name: Khateeb Ahmad

Programme: BSc. (Hons.) in Computing

Month: January 2018

My Achievements

This month, I was able to complete the entire UI for my application. I created all pages from my mocks and was also able to test them using smoke tests. By creating all pages, I have put a final shape to my application and this would also ensure that I don't implement something that was not original a planned part of my application.

My Reflection

This month went well as I was able to complete UI and it gave me a real sense of accomplishment and also gave me a sense of direction as to how my application would look like and behave. This meant that I was able to translate design from my mind and mock-ups to actual prototype and could plan and make changes according to better fit the use cases now.

I would have liked to complete some functionality as well but was not able to do so. Even completing a simple feature like adding service providers would have put my project development on real pace.

Intended Changes

For the next month, I will try to complete all basic functionalities like logins, adding providers, searching them, reviews. By the end of next month, the goal would be to finish a basic version of the product and complete its testing as well.

Supervisor Meetings

Date of Meeting: 23 Jan 2018

Reflective Journal

Student name: Khateeb Ahmad

Programme: BSc. (Hons.) in Computing

Month: February 2018

My Achievements

This month I completed basic functionalities to be supported by my application. I completed logins and user profiles for both service providers and customers. Then, I completed registration for a provider. After that I made searching functionality where a user can search a provider based on the location. Once a provider had been selected, I implemented the code to book them for appointments. I also implemented the functionality to provide reviews for service providers.

My Reflection

This month was very good as I was able to complete a basic version of my entire portal. It helped me to have a UI for my application ready which meant that I could provide functionalities around and link them up easily. It also meant that it did not take much time for me to jump from one functionality to another.

However, I was not successful in testing my portal using various testing techniques. I was only able to write unit tests for all functionalities I implemented. But, I wanted to test with smoke tests and maybe automated tests but was not able to come up with them by the end of the month.

Intended Changes

Next month, I will try to implement all the differentiating functionalities like Facebook integration, Location based search, Semantic web and robotic process automation. I will also try to come up with automated tests for my entire portal.

I realised that I need to also integrate PayPal sandbox for payments. This requirement is something that came up after discussion with my supervisor and I will take this one up immediately.

Supervisor Meetings

Date of Meeting: 12 Feb 2018

Reflective Journal

Student name: Khateeb Ahmad

Programme: BSc. (Hons.) in Computing

Month: March 2018

My Achievements

This month I started to implement functionalities that differentiate my project from others and started to give my idea final shape. First, I implemented Facebook integration for sorting friend reviews on the basis of reviews given by friends. Then, I implemented location based search where the location of the user gets picked automatically and then the user can continue the search. Then I implemented some portion of the functionality where a service provider would get a rating based on sentiment using semantic web.

My Reflection

This month was good again as I was able to get some exciting functionalities implemented and finally my project has started to reach its final stage. The functionalities were tested extensively as well and I was happy as I did not encounter any major bugs.

I would have liked to finish semantic web implementation but was not able to do so as I was busy in project for other modules as well.

Intended Changes

For next month, completing semantic web implementation and integrating PayPal payment gateway will on top priority. If I am sable to successfully complete them, only then I will move to robotic process automation.

Supervisor Meetings

Date of Meeting: 12 Mar 2018

Reflective Journal

Student name: Khateeb Ahmad

Programme: BSc. (Hons.) in Computing

Month: April 2018

My Achievements

This month, I completed semantic web implementation and was also able to make PayPal payment I was able to make some structure for robotic process automation. I then favoured the stability and reliability of my portal and spent time in testing it.

My Reflection

This month was particularly hectic as I was trying to finish my project and had project in other modules as well. This made time allocation very difficult. But I also felt that by planning early, I had reduced a lot of discovery part which meant less surprises and so the project development was smooth.

Supervisor Meetings

Date of Meeting: 2 Apr 2018