# Online Diary Web Application

# 1. Introduction

The developed system is an Online Diary system, which allows users to share their contact details and schedule appointments with other users around the world. This report describes the development process of this system.

# 2. Design

After analyzing the requirements of the system, I have decided to divide the system into the following views:

* Account creation and login
  + The pages which will allow the user to create an account and log in to the system
* Home page
  + The page which will display the list of all the users, with a search functionality
* Appointments
  + The page which will display a list of all the appointments for the logged user
* Booking appointments
  + The page which will allow the user to book appointments with other users
* Profile
  + The page which will allow the user to modify their details

The application is going to use material design, as the styling and colour schemes are well known to the users. This will increase the usability of the system and bring modern design and look. The users will navigate through the application by a navigation bar at the top of the page.

All the error messages displayed to the user will be shown under the input which has caused an error with a red colour. This is to make sure that the user understands the error message and where did the error happen.

The key components of the application:

* Entity classes

1. The Person entity class will hold all the information specific to the person such as names, username, email etc. Additionally, it will contain the related address for this user using Address class, and its list of appointments using the Appointment class.
2. The Address entity class will hold all the address information such as street, city, country etc.
3. The Appointment entity class will hold all the appointment details, such as time, date, and guests.

* Controllers

1. PersonController will be responsible for the management with everything related to the Person entity class (Creating a person, login, displaying a person list).
2. AppointmentController will be responsible for the management of the appointments, used in appointments screens to show all appointments for the logged user, and all the methods for creating appointments.
3. NavigationController will be used to generate the links in the navigation bar, additionally different colour will be displayed on the currently active link.

### 2.1. User interface design

Below are low prototype user interface designs which will be used as a blueprint when it comes to implementation of the system.

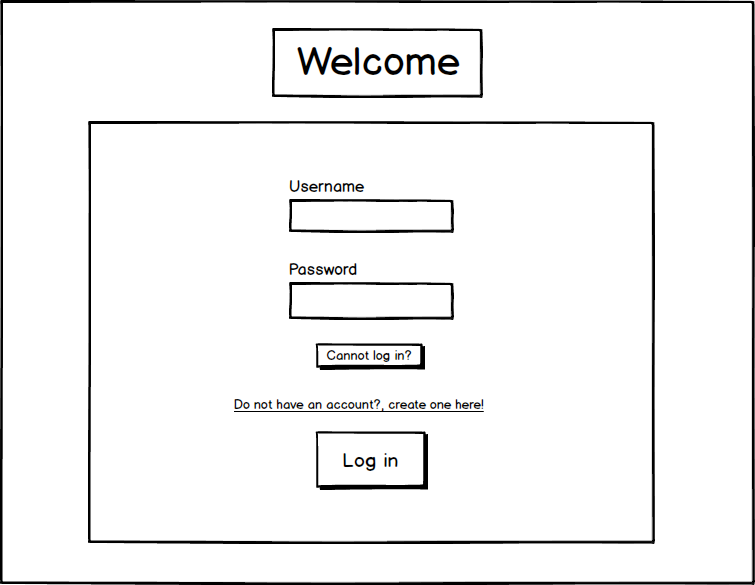


Figure 1. Login screen

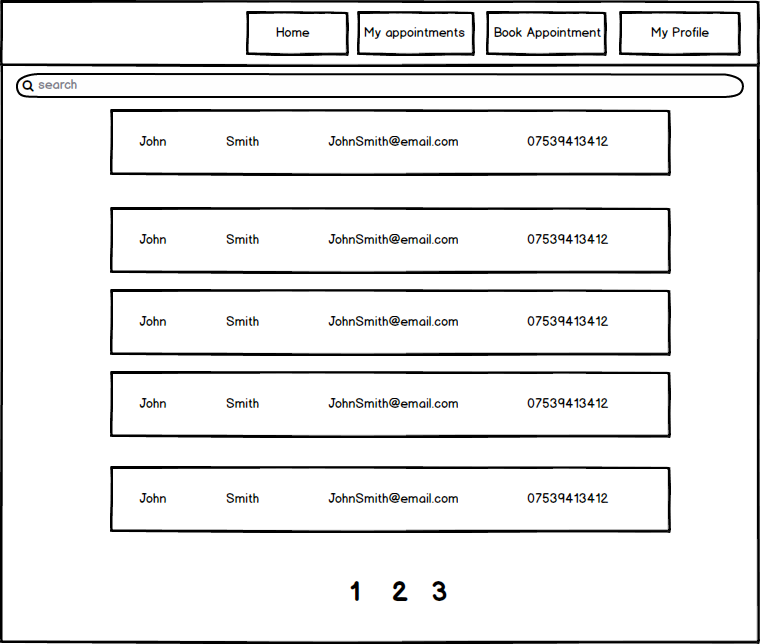


Figure 2. Home screen

### 2.2. The flow diagram of the system

For the user to use the system, he must create an account and log in. Then he will be redirected to the home page of the application. From there a list of all users will be paginated and displayed. After login, the user can navigate through the app using the navigation bar located at the top of the page. The available options will be Home, Appointments, Book Appointment, Profile and Sign out.

# 3.Implementation

## 3.1. Development tools

Java Enterprise Edition 7, Java Server Faces (JSF), Java DB, Glassfish 4.1 and NetBeans IDE were the development tools used to develop the system. I have decided to use NetBeans because it supports the development of applications with the frameworks specified above. It was not easy at the start to use the IDE, as I had no previous experience, however, the use of code generation mechanics, such as getters and setters, together with the wizards to create different classes for facades and beans made the development much easier. Additionally, Version control was used to manage the project, which allowed to easily keep the commit and merge history.

## 3.2. Development

Before starting the development of all the functional requirements, I have set up a correct project structure and creates all the necessary project files. Following the design, I have created all the entity classes, controllers, services and facades. At last, XHTML files were created for each of the screens mentioned in the design.

To reduce duplicated code and improve maintenance of the project, research was done to create a template for the most common code such as the header and the navigation bar. With the use of **ui: composition**, additional pages were created which were used as a template for all other pages. MaterializeCSS library was used to style all the elements such as buttons, inputs and text with my own separate style sheet to position all elements in correct places. With entity classes it was easy to insert, update and delete the records from the database. Any time I had to make a change in the database, doing a change on the entity class automatically updated the record, which sped up the development of this project.

## 3.3. Issues

The first problem which I have encountered was when I was developing the login system. Upon successful login to the system, the user had to be redirected to the home page of the application. The user was getting redirected but none of the data was there and the current logged user variable located in the PersonController was null. This problem was resolved after setting the scope of the PersonController to SessionScoped instead of previously used RequestScoped. Another issue which I have encountered was that upon a deployment of the application all the data inside of the database would get pruned. This made it very difficult to test, as I had to go through the registration/login every time I have made a change. Even changing the configuration of the database in persistance.xml to CREATE, would not resolve the issue. This issue was resolved after recreating a new database and creating a new data source.

I was encountering a lot of errors when developing the software, at first, I would try to check the logs and try to find the error messages, sometimes I would try to print the values in the logs, but the best approach to this was to use a debug functionality in the NetBeans IDE. Placing a breakpoint allowed to slowly move through the code line by line, additionally all the variables were visualized, therefore an error could be easily spotted. This has helped a lot in the implementation as I knew exactly where the error happens and why.

## 4. Testing

Each of the functional requirement was tested manually. After implementing a requirement, manual tests were performed to check if the requirement is met. Testing classes were not used; therefore, it is hard to prove that all the functionality works as expected. This is due to a time constraint.

## 5. Summary

I have successfully completed the application in a framework which I have not used before. I have followed the requirements and design specifications. I am happy with the outcome of the application. In terms of a design of the page it looks modern, if compared to any other web applications. With the use of AJAX in JSF, the application has no reloads and all the data is automatically displayed to the user. The application handles the validation well, the messages are shown under each input which has not passed the validation, helping the user to correct the errors.

Even though I am happy with the look and feel of the website, if I was implementing this again, I would not use any external CSS framework. It would be much easier to use a library specific for JSF such as Prime Faces, as the components already have the defined styles. The application would be better, if I had used a proper testing approach such as Junit instead of manually testing.

### **References**

Briggs, J. (2019). Step by step JPA/JSF webapp. Retrieved from http://briggs.myweb.port.ac.uk/WEBP/notes/db/jpasteps.html

Getting Started - Materialize. (2019). Retrieved from <https://materializecss.com/getting-started.html>

JSF Managed Beans. (2019). Retrieved from <https://www.tutorialspoint.com/jsf/jsf_managed_beans.htm>

JSF UI Components - javatpoint. (2019). Retrieved from <https://www.javatpoint.com/jsf-ui-components>

Lars Vogel, 2. (2019). JSF (JavaServer Faces) - Tutorial. Retrieved from <https://www.vogella.com/tutorials/JavaServerFaces/article.html>