**Milestone 1:** Login and Signup service to end user

**Time**: 100 hrs.

**Sub-milestone**: Providing login service as UI layer

1. Login Page Description

The Login page is created which will be displayed as soon as someone (user) wants to book the flight if the user not already logged in. The Login page is mainly for accepting and validating user ID and Password by checking the user credential with those stored data-base. If user exists he/she is redirected back to the home page as a logged in user and he/she can start his/her bookings. If the user first selects the flight, he/she is asked to log in before proceeding with the bookings. If the user does not exist, he is asked to create an account.

**Stories**

* **User ID**: The user ID should accept and validate the Email-ID of the user.
* **Password**: The password should accept the user inputs which should be minimum of 8 character and maximum of 20 character.
* **Login Button:** The Login button should check whether the user ID and Password will match if match then it should redirect to home page ,if not it should show message that is “Invalid Username or Password”, if he/she is a new user it should show a message that is “if new user Sign Up”.
* **Sign up:** For the new user if he clicks ‘Sign Up’ it should be redirected to the Sign Up page.
* **Typescript component:** This component will make rest call to backend (server).

**Sub-milestone**: Providing signup service as UI layer

2. Sign-up Page Description:

If the user has account, user can login directly. If user don’t have account user will be redirected to sign-up page. Sign-up page require following details to create an account, i.e username, password, confirm password, email-id, contact number and confirmation button, followed by Validation against input requirements, stored in database. Two validations are carried out – one to check if the password and confirm password data match or not and second to check if the input data is according to the requirements set. The user is redirected to the home page as a logged in user.

**Stories:**

* **Data:** Registration page can include username, email, contact number, password, confirm password.
* **Validation:** Built-in mechanisms which validate the quality of the data (e.g. email format), password strength, matching password and confirm password and empty input.
* **CSS and HTML:** A more advanced option is to edit the screen-set’s CSS files, or the HTML.
* Use a <form> element to process the input.
* Use <label> and <input> to create username, password, confirm password, email and contact number.
* By creating an account agree for terms and conditions, use <a> tag for terms and conditions.
* Use a <button> to sign-up or cancel the page.
* Already have an account login.
* **Typescript component:** This component will make rest call to backend (server).

**Sub-Milestone** : Provide Login-Registration service as a Business Layer.

**Description:** Here we are validating the user on basis of his username and password and if he is a new user, he is allowed to register.

**Stories**:

* **Login Service :** After the user logs in using his credentials, the username and password will be verified with the existing database using isValid(String username, String password) method.
* **Option**: If the user has forgot his/her password and wants to change the password, the forgotPassword (String email) method allows him to change the password after checking if he is a valid user.

* **Registration Service** **:** The user who is new to the website will be redirected to the registration page once he clicks sign up button. The addUser(User user) method provides him the service to get his credentials added into the existing database

**Sub Milestone** : Login and Registration Controller to provide RESTful services as a Business layer.

**Stories:**

* In LoginRegisterController class all services of login service and register service classes are mapped using @RequestMapping.
* For adduser(User user) method of register class, mapping is done with @RequestMapping(method=RequestMethod.POST, value=”user”) , after successfully adding to the database he/she will be redirected to login page.
* For isValid(String username, String password) method of login service class, mapping is done using @RequestMapping(“/login”) and if it is valid, he/she will be redirected to the home page.
* For forgotPassword(String username) method of login service class, mapping is done using @RequsetMapping(“/forgotpassword”) and after successfully changing password, he/she will be able to login using the new password.

**Sub-milestone**: Providing login and signup service as business layer

**Story**: Users class

**Description**: User class is used as entity for the database. This will help for object relational mapping with database.

**Sub-stories**:

1. Instance variables of Users class

-userEmail- string

-userName- string

-userContact-long

-userPassword-string

1. Constructors

-default constructor

-parameterisedconstructor with arguments userEmail, userName, userContact, userPassword

1. Getter methods

-getUserEmail()

-getUserName()

-getUserPassword()

-getUserContact()

1. Setter methods

-setUserEmail()

-setUserName()

-setUserPassword()

-setUserContact()

1. Make User class as @Entity

Making email as @Id

**Sub-milestone**: Providing login and signup service as database as service

**Story**: Users collection

**Description**: User collection will store the data related to user (username, useremail, usercontact, password). This will provide restful service to business layer.

**Sub-stories**:

Users collection- Fields-useremail(primary key), username, usercontact, password.

7