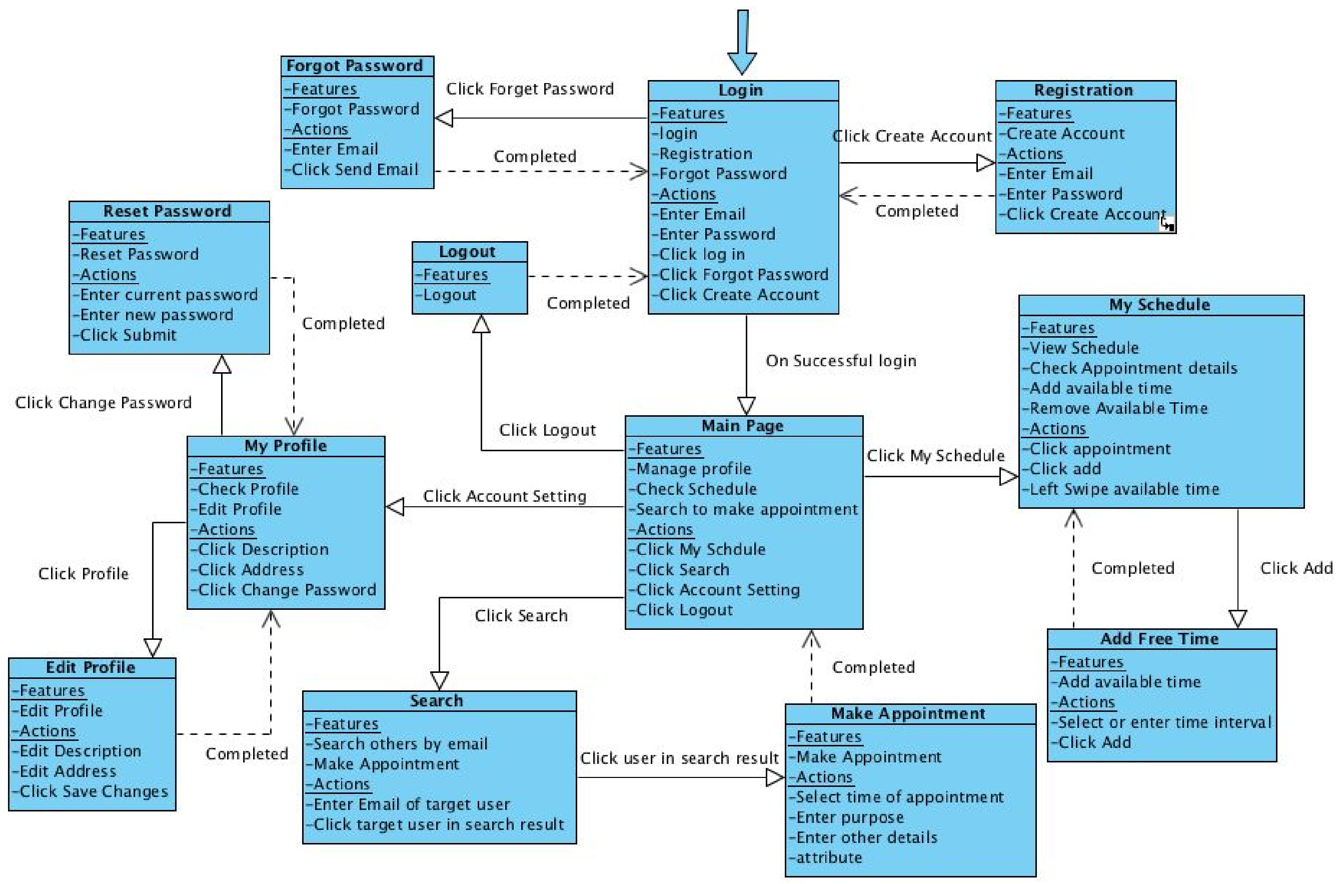
Tendee

Team 9

Incremental Testing & Regression Testing

----Chaolun Shao, Altug Gemalmaz, Avinash Prabhakar, Chau Minh Nguyen

---



1. Classification of components

1.1 Module description

Registration Module

Module in charge of creating a User and adding his email, name, description, address to the firebase database. A connection is established when the create account button is clicked and the data is added to the Users table.

Inputs: email and password. In the next page, name, description and address.

Outputs: Moves to main page as confirmation message that user has logged in.

Login Module

Login module is in charge of identifying who the user is and according to the provided user information contacts the authentication to get the information of the user.

Inputs: Combination of email and password of the user.

Outputs: The main UI of the user if the combination is valid, a warning message if the combination is not valid..

Logout Module

Module in charge of logging out a user. Verification is done to check if the user is already logged in or not. When logout button is pressed, Firebase authentication establishes whether user is logged in and logs him out..

Inputs: Logout button pressed.

Outputs: Confirmation by the User Interface as user is sent to start page.

Edit Profile Module

This module is in charge of managing user personal information. When this is done, this data is sent to the database for storing. When a user desires to change his/her personal information again this module is responsible for getting the information from the user and sending/updating the information on database.

Inputs: User data for example (email, password, name, description and address)

Outputs: If the user data is entered correctly, the new data will be sent to the database and a confirmation text will be displayed informing user that the data is updated.

Forgot Password Module

This module is in charge of resetting the password of the user if that user forgets it. In the start activity page, once the user presses the forget password button and enters his or her email, a connection to Firebase is set. Firebase Authentication then sends a email to the user. The user can then click the hyperlink in the email.This opens up a browser and the user can enter a new password. Password is then updated.

Inputs: Pressing forget password button which opens up a dialog box. Enter email address in text field.

Outputs: Dialog box closes and Firebase Authentication sends an email to the email address of the user.

Reset Password Module

This module is in charge of changing the password by using the current password. This module is solely responsible for handling password data because this module has to be secure to protect user privacy. Once the user presses the change password button and enters the valid password, Firebase Authentication updates the password for that user.

Inputs: 1. Current password. 2. New password. (Entered twice.)

Output: A confirmation the password were changed successfully or a warning on invalid password.

Main Page Module(Sprint 2)

This module is in charge of navigating user to different modules like “My Schedule”,“Search”, “My Profile” and “Logout”.

Input: Button pressed

Output: Navigating to correct module.

My Schedule Module (Sprint 2)

This module is in charge of checking the schedule of user. There are two different kinds of categories in the schedule list: available time and appointment time. User can check an appointment by clicking this appointment and the details will be displayed.

Input: Items select/Button Press

Output: Details of appointment

Add Available Time Module(Sprint 2)

This module is in charge of adding available time to the schedule so that other user can see it. User can select a time interval and mark it as available.

Input: Time interval (start time and end time)

Output: A confirmation of whether or not the time interval was added successfully

Search Module(Sprint 2)

This module is in charge of searching other users in this application. User can use email to find the profile and the schedule of other users.

Input: The email of target user

Output: The profile of target user.

Make Appointment Module(Sprint 2)

This module is in charge of making appointment with user that found through searching. User can make appointment with other user by selecting an available time and filling other details(attendee, purpose and phone)

Input: Select time and enter details.

Output:A confirmation of whether or not the appointment was scheduled successfully

Interface Component - Button

The Button widget is used to create selectable buttons in the user interface. These can be customized in terms of size, color and type etc.

Inputs: Button Id with details like Button text, color and onclicklistener function.

Output: On click response is called when button is pressed.

Interface Component - Text Field

The TextInputEditText widget is used to create editable text fields where users can enter text. These can be customized in terms of text size, positioning and width of field etc.

Inputs: TextInputEditText Id with details like text hint and listener attached to a button.

Output: Returns a string when the listener detects the button has been pressed.

1.2 Incremental testing technique

Right now, we are using top-down testing technique. We start with testing I/O by confirming user input to sign up and log in text input, then test if interface elements such as button, navigation bar works. Then we move on to test the functionality of sign up, which is a prerequisite for sign in. We think this is suitable for us because this can reduce the use of drivers or simulation of fake data. This also saves time as we can ensure each component works first before moving on to the next one.

2. Incremental and regression testing

2.1 Description

Registration

Using Android JUnit 4 testing framework with Java, we continue testing some edge cases regarding the information user provided when they sign up, such as their email, password, name etc. We also simulate some error input for regression testing from previous tests.

Login

Login module uses Android JUnit 4 testing framework to simulate different inputs and interaction between user and application. Also we test some cases where internet connection is not available.

Logout

Logout module uses Android JUnit 4 testing framework to simulate different states and interaction between user and application

Edit profile

Edit profile module uses Android JUnit 4 testing framework. Tests are incremental from sign up module to make sure information is persistent from when user sign up. Some are regression test to fix our bugs from last development time.

Forgot password

Forgot password module uses Android JUnit 4 testing framework. This includes incremental tests from sign up module and regression test to verify previous bugs have been fixed.

Reset password

This module was testing using Android JUnit 4 testing framework. Tests are designed to cover various inputs, including different combinations of the components.

Interface components

Components like Text Field, Button widgets are tested using Espresso which is a User Interface testing framework. Tests cover many edge cases and various input.

2.2 Defect Log

Module	**Registration**		
Defect Number	**Description**	**Severity**	**Solution**
1	Button interface fixes on the main UI (So that the links between the UI's could be established) was causing some UI's connection (Log in to main UI connection) to break.	1	Instead of automatically generating the button interface links between the UI's we manually found the links (references to the UI's) and manually entered the UI's references to the variables that was in the XML. We gave unique IDs to each UI component which fixed the problem.
2	The user data that was entered when tried to submit all the data at once caused a transmission exception to the database. We figured out that this was a FireBase related defect.	1	We changed our way of sending all the data to the base. Instead of sending all the data by itself, we decided to use a HashMap and send the data as a collection.
3	When the registration was done the user "supposedly" had to receive a confirmation email which was not working.	2	In order to solve this problem we had to first send the user data to the database. Later on let the execution continue (on the database side) until it's ready. Then utilize the confirmation email service which is provided by FireBase. By doing this we managed to avoid compatibility issues and also use the easy service that was provided by FireBase.

Module	**Login**		
Defect Number	**Description**	**Severity**	**Solution**
1	User can login without email verification. This happens when user just finishes registering and being redirected to login activity.	2	We prevent the user from immediately logging in once he creates an account. This prevents him or her from logging in without verifying his or her email. By using Firebase API, we have a conditional check to see if the user has verified his email before he or she tried to log in through the login page.
2	An error is thrown when the user tries to login without creating an account. The User Interface will show no response and the user is not able to see what went wrong.	2	Using Firebase Authentication API, we were able to use a built in method which will check the credentials that the user has input. It validates whether the credentials are correct and if it point to an account that already exists.
3	An error is thrown when the user tries to login with incorrect email or password. The User Interface shows no response so user does not know what went wrong	2	Firebase Authentication provides an API which has a method that checks if the email password combination is correct. If it is, it returns true. We used that method to fix the problem.

Module	**Logout**		
Defect Number	**Description**	**Severity**	**Solution**
1	The app does not return to the start page when the user click the logout button. The user is unable to sign out of his or her account, which would cause him or her to be stuck in the app.	1	Firebase Authentication provides an API which has a method which set the current user to be null, which effectively signs user out. By using that method, we check if the current user is null. If it is, we go back to the login page.

Module	**Edit profile**		
Defect Number	**Description**	**Severity**	**Solution**
1	If the string of data is too long, the text will overflow on the User Interface when pushing the data to the app. This makes it hard to see what is on the screen.	3	We set a limit to the amount of text that can be displayed on the User Interface. If any text field is too long, it is truncated by setting the truncation attribute of the text field.
2	If there is no internet access, no data will be loaded on the app. The app did not raise any error message when this happened. So the user will not know why the data is not loaded.	2	We check if the app has internet access, if there is not, a message is printed on the screen informing the user that he is not connected to the internet when he enters the profile page.

Module	**Forgot password**		
Defect Number	**Description**	**Severity**	**Solution**
1	When the user clicked the forgot password button, entered email with incorrect format.	1	We change the attribute of the text field to accept only text which has the correct email format or it would throw an error.

Module	Reset password		
Defect Number	**Description**	**Severity**	**Solution**
1	Resetted password even if current password entered is incorrect.	1	Using Firebase Authentication API methods, we verify if the current password entered is identical of the current user. If it is, only then proceed with the password change.

Module	Button UI component		
Defect Number	**Description**	**Severity**	**Solution**
1	No error messages were raised when onClickListener function was missing or not triggering. The User Interface lets you click the button but nothing would happen.	3	Clicking a button with no onClickListener will now raise errors stating that this button will do nothing since it has no onClick functionality.

Module	TextField UI component		
Defect Number	**Description**	**Severity**	**Solution**
1	When the listener detects that a button has been pressed, no error messages show up if the textfield returned null. This happens if the user did not enter any text in the textfield	1	We put a check to see if the text field returns null. If it is null, The variable pointing to it should be set to hold an empty string instead of null.