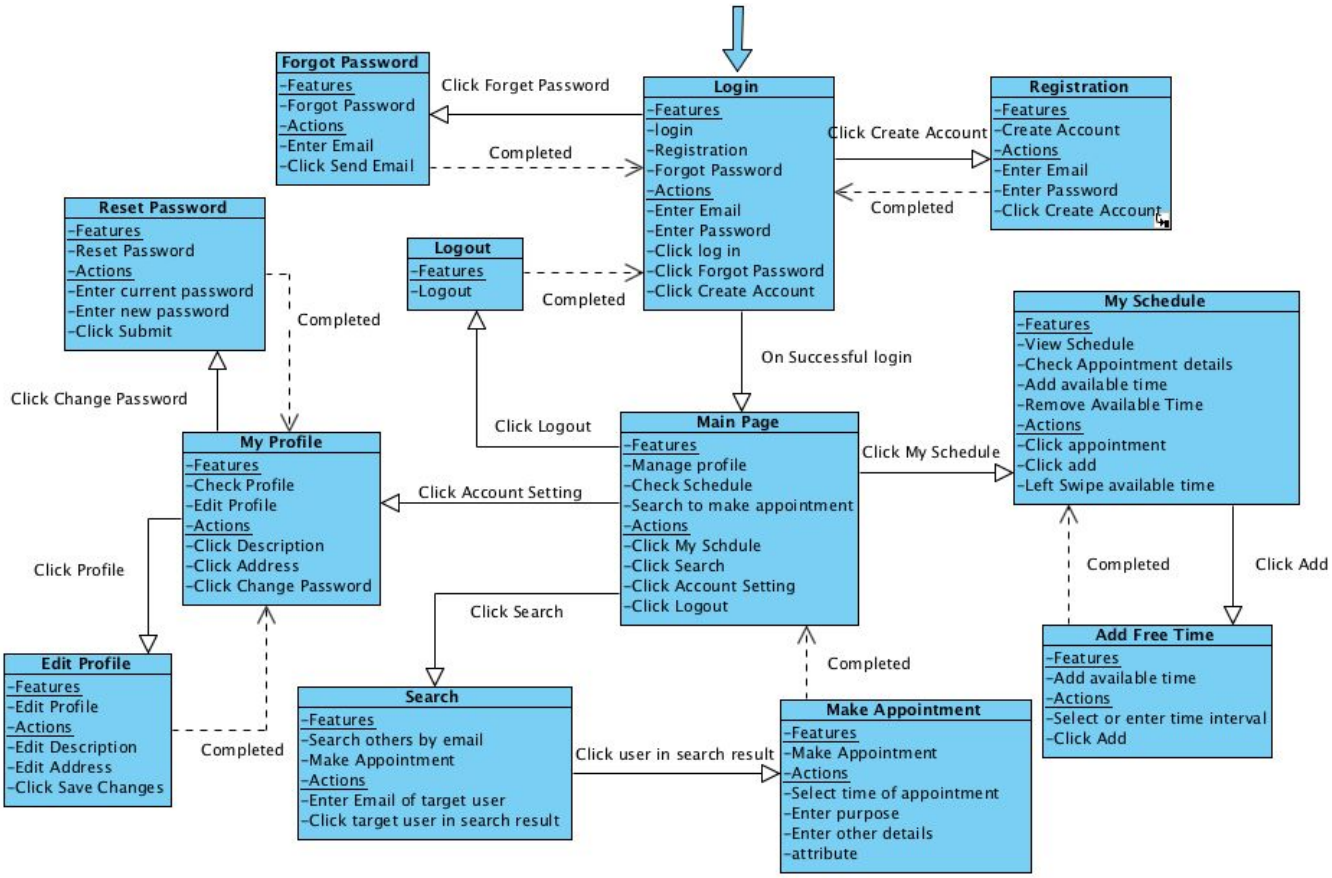
Tendee

Team 9

Sprint 2 Incremental Testing & Regression Testing

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

---



1. Classification of components

1.1 Module description

Schedule Manager

Schedule Manager deals with schedule of the user, such as creating, updating, removing, and viewing available time slot of local user.

Inputs: Time slot to be added/updated
Outputs: Display updated schedule of user or current schedule if user just want to view

Search Module

Search Module allows user to search for other users in order to make an appointment

Inputs: Name or email of other users
Outputs: List of users

Appointment Manager

Appointment Manager allows user to make appointments with other user.

Inputs: time slot to book an appointment
Outputs: confirmation that booking appointment was successfully.

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

Schedule Manager

Module testing was done in method used to create available time slots, printing success and failure. The tests cover edge cases and unexpected input. Also JUnit is used for the functional test cases to be tested to make sure that they function. Espresso framework is used to see when the sequential appropriate steps taken by the user, the app functions correctly. FireBase (Backend) side is tested with the help of Espresso, backend data validation is handled with Espresso.

Search Manager

Search Manager was done in method to search for other users. Test are designed to cover null cases and variety of input when searching for other users. Junit and Espresso is used to automate the test cases. Junit creates automated test cases to see if the inner components function correctly. Espresso tests from the user perspective, how the user would take steps to use this app.

Appointment Manager

Appointment Module was done in method to create appointment. The tests included incremental testing from lower-level module and regression tests to make sure bugs from lower-level does not exist. Espresso and Junit is used again for this module to automate this process.

2.2 Defect Log

Module	**Schedule Manager (Incremental Testing)**		
Defect Number	**Description**	**Severity**	**Solution**
1	User should only create available time between 9 am - 5 pm	2	Put boundary checker to check the user input
2	User should enter minute as 30 or 0 minute because something else will create bug in database	2	If the user choose a time greater than 30 minutes , it will be rounded up.

			If it is lesser than 30, it will be rounded down.
3	During getting the appointment input from the user there was a multi-threading issue that was causing app to crash.	1	Make the current thread sleep for a 100 ms for each instruction.
4	When user marked a time interval as free, sometime it can not be recorded in user's schedule completely and it will miss half hour	2	Rewrite the algorithm to convert time interval to half hours blocks and modified the way to store time interval.
5	When user marked a time interval as busy, it may sometimes remove the free time incorrectly.	3	Rewrite the set busy time interval algorithm to to have stricter boundaries so that it does not go beyond the time limit.
6	When the the user marked a time interval as busy followed immediately by free and then busy again, the time interval might not be printed correctly even if the backend code is correct.	2	Changed the way time intervals are displayed on the app. Keep two pointers that keep track of the starting and endings of the free time. Using the pointers, the time intervals would be more accurate.

Module	**Schedule Manager (Regression Testing)**		
Defect Number	**Description**	**Severity**	**Solution**
1	Fixing SettingsActivity.java to store the local changes when someone makes a change in the settings, was causing data inconsistency in the database communication module. This was leading to inconsistencies between the database and the app.	1	Instead of generating a local object to store the local changes we decided to send the changes immediately to the database. Through this we achieved consistencies between the app and the database.

2	Fixing the schedule incomplete problem, was causing the duplicated time blocks.	1	Instead of using one loop to access both free list and appointments,used two separate **for loop** to go through the schedule list and display free time and appointment.
3	Fixing the schedule module multi-threading, so that more tasks will be done, was breaking data consistencies between different sets of modules.	2	Make the critical section code to be protected by enabling only one thread to access.
4	Fixing creating free time, the database should change data attribute to 0.	1	Use Firebase query to find the appropriate attribute and change it to 0 when time is free.
5.	Fixing the schedule module appointment update problem, sometimes caused no appointment number to be pushed into schedule table.	1	Instead of only listening for a single event, now it is always listening for a data change. This ensure when a appointment is created, the appointment number will always be pushed to the schedule table.
6	Fixing the schedule module where free time intervals are being split up and duplicated.	2	Use another variable to keep track of the position where the each free time interval ends. This variable would be used to ensure that time intervals are not being duplicated.

Module	**Search Manager (Incremental Testing)**		
Defect Number	**Description**	**Severity**	**Solution**
1	When search does not have any input, app should display no result.	3	After query list name return, check size of list name, if list is empty, display no result.
2	When search bar input is not-empty and user presses search, this displays every users.	2	Check the length of input before searching for users, if the length is less than 2, the search will not be

			processed.
3	When user enters text too long, causing part of text be hidden.	2	A limit is added to how long the text can be.
4	User searching function so that it gets the desired user from the database was crashing the search bar.	2	Instead of getting all the user information make the database send only the desired user information.
5	When user choose not to add more attendee, the app crash when display the appointment details.	1	Instead of displaying all of attendees separately, combine them into a attendee list.
6	When the user X is blocked by user Y, the block switch will stay off when opening the profile of the user X.	2	Before loading the profile a user X, check the block condition and change the default status of switch.

Module	**Search Manager (Regression Testing)**		
Defect Number	**Description**	**Severity**	**Solution**
1	Fixing the user info getter buffer to a big size to handle multiple users was making user buffer to allocate too much buffer even for a single user. Consequently, it was allocating too much memory and make the other modules stack overflow.	3	Instead of always assuming there will be multiple users, allocate the buffer according to the number of data that is received. For a user that is received increase the buffer size.
2	Fixing the user getter to get multiple user data at once through concurrency was making the shared resources to be corrupted, consequently creating complications for the modules that was working with the user	2	Make the user data getter to sequentially get the user data instead of using concurrency to have the data quickly.

	data.		
3	Fixing user getter module to get every user when a simple input was typed was slowing down the app and also was breaking the UI format.	3	Instead of making the user getter to start getting the user data even when there is few input, we decided to put a checker and when the total character count was higher than the desired count then the user getter module was getting the user data.
4	Fix the app crashed when other attendee is exited causing the other attendee can not see this appointment in his or her schedule when there are two attendees,	1	Instead of store the attendees as a attendee list, store them separately but still display them as a list and initialized the second attendee as " " and add more checker for "Only one attendee" situation
5	Fixing the display of search result, when there is no result, it should display "no result".	3	Check empty result from query, then display accordingly.
6	Fixing the user detail to show up when clicked, was causing app to crash because the user detail was not loaded when the user name appeared on the bar. Consequently, fixing the search bar to display users was crashing the search module.	2	When user clicks on user info make the execution sleep for 100 ms and meanwhile make another thread to fetch the user profile data from the database.
7	Fixing the **off status** of switch when opening the profile causing the app keep adding this user to block list and remove it from the list at the same time.	2	Instead of using addValueEventListener we used addListenerForSingleValueEvent then it will not keep adding and removing at the same time.

Module	**Appointment Manager (Incremental Testing)**		
Defect Number	**Description**	**Severity**	**Solution**

1	The appointment that is created should be displayed to all of the users who are in the appointment but some of them are missing	1	The appointments are not stored locally on the phone data instead each created appointment is stored on the database globally.
2	A change that is made to the appointment is only displayed to current user instead of all the users in the appointment	2	The appointments are regularly updated from the database to ensure that the appointments are up to date.
3	Appointments should be created according to time module handler. The time checker module should make sure that the appointments times are appropriate.	2	Call the inner functions of time module and pass the values, if the parameters are acceptable then update or create the appointment.
4	Sending email to user was working properly without error but user can not receive email.	3	Create a new thread to run the sending email function instead of using the UI thread.
5	User can make appointment with other user even though the schedule is not available(e.g make appointment with someone on Monday but it is already Tuesday).	2	Added a time filter to remove all of the expired free time and appointment.
6	Making appointment of time between 2 days like 12 pm and 1am seems to cause only 1 day to change.	2	Put a time checker so people can only make appointment between 9am and 5pm.

Module	**Appointment Manager (Regression Testing)**		
Defect Number	**Description**	**Severity**	**Solution**
1	Fixing the sending email feature would cause user receive 7 emails at a time when made an appointment and the ui will not go back to the main activity	3	Create a new java activity for sending email only and call the sending email function in oncreate()method and still use a new thread

2	Fixing the data overflow by simply truncating the data would cause the app to crash eventually as the data is still stored in the database.	3	We made sure the database would not hold data that is too long by ensuring that for every variable there is a character limit to how long the string value is.
3	Fixing appointment holder to have multiple appointments was causing issues at the appointment displayer module. The appointments couldn't be able to seen on the UI.	2	We decided to put the received appointments in a array according to their upcoming time. Then display the appointments next to one another as a whole schedule.
4	Fixing the appointment module to include too much user text, was breaking the UI, the format was being lost.	3	We put a word limit to the total number of characters that can be inputted from the user.
5	Fixing the input checker to check every input that was entered was crashing the entire module. Because it was also trying to check even when the input was NULL.	2	We made sure that when the input was NULL, the input checkers was not being called.