Tendee

Team 9

Design & Code Inspection & Unit Testing

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

---

Design Inspection

Product	Design Inspection		
Date	02/07/2018		
Author	Tendee Team		
Moderator	Avinash Prabhakar, Chau Minh Nguyen		
Inspectors	Chaolun Shao, Altug Gemalmaz		
Recorder	Chaolun Shao, Altug Gemalmaz		
Defect #	**Description**	**Severity**	**How Corrected**
1	Password is entirely displayed in the text box when entered it.	3	Each character of password will be hidden behind asterisks after next character is typed
2	On login screen when a non-existent user info entered error occurrence.	2	By handling the error, instead of letting the exception occur we handled this case by printing “no user found”.

3	Even though the user didn't log in, the program was taking user to the main view which should be only displayed only if the user was identified.	1	By first doing user authentication, we checked if the user was identified in our system. Only if the user was in our system, we enabled access to main view.
4	When the user clicked "send email" button during registration, the user would still receive email even though the user has registered with the same email. It is not appropriate to receive a registration email again if the email is already on database.	2	We Check the existence of the user and his/her email before sending verification email. If the user has registered, we pop up a warning.
5	Amazon Web Service didn't provide most of the functionalities we wanted to have and created compatibility issues.	1	We switched to FireBase which provided us the services that we wanted to have. Also we managed to solve our compatibility issues with our new database.
6	On the log-in UI, either when the username or password was not entered (it was NULL) and when this information was sent to the database, there was an exception.	1	Entered input is now first checked to see if it contains a value (not NULL). If it didn't have the required (its NULL) inputs, the function call to the database is now avoided. Now simply the UI doesn't do anything if there is no correct input.
7	In the log-in UI when tapped onto the back button it didn't switch back to the previous UI.	2	By entering the "parentActivityName" variable it's right value, we managed to fix this issue.
8	User's confirmation code will also be stored in the database at all times. A very important bottleneck that we realized as the number	3	A very simple solution that we come up with is to periodically remove old user-confirmation pair at regular time intervals. If this is done every minute every process, task, will be slower. We can trade-off a little bit memory to

	of users increases, registration process will be slower.		efficiency.
9	The password could have been entered even 1000 lines. This was crashing everything. A very simple strategy for a malevolent user to crash our system.	1	By checking the length of the password, if it was very long we stopped the execution and asked user to enter a shorter password. Until the shorter password (less than 20 characters) was entered the user can't continue to the next step. Simple now the UI doesn't do anything until the inputs are valid.
10	When the internet was down and the user tried to login there was expectation. Since our application is very internet dependent because it talks to the database frequently this was a big issue.	2	This is fixed by checking the internet connectivity first before trying to connect with the database. If the internet connection wasn't established then a warning string is printed on UI screen.
11	When clicked on the "forgot password" button, user will receive his or her password by email. This can cause security issues.	3	Send an email that contains a temporary password instead of the passwords itself. User can use this temporary password to login for once and then he or she can enter another password.
12	User can set password as "12345678"	3	For security reasons and to make sure passwords are not predictable, add a password checker to make sure the password contains number and alphabet.

Code Inspection

Product	Code Inspection		
Date	02/07/2018		
Author	Tendee Team		
Moderator	Chaolun Shao, Altug Gemalmaz		
Inspectors	Avinash Prabhakar, Chau Minh Nguyen		
Recorder	Avinash Prabhakar, Chau Minh Nguyen		
Defect #	**Description**	**Severity**	**How Corrected**
1	FireBaseAuth.getInstance() might return null	1	check null before use mAuth
2	register_user() method is public	3	make it private
3	Lack finish() function at the end of register activity	2	put finish() after redirecting user to main activity
4	email, password should be both filled before continue	3	replace OR with AND operator
5	getSupportActionBar() may return null	3	use a variable as a return from getSupportActionBar(), then check null

Unit Testing Report

Framework: We used the Espresso library for our project because Espresso is an automation testing library created by Google and made specifically for Android. Espresso is used primarily for black-box testing, however Google recommends having a working knowledge of the codebase to get the full effect of Espresso Tests. And Espresso is targeted at developers, who believe that automated testing is an integral part of the development lifecycle. This method is better since our Firebase connection is separated by each screen. In addition, the online tutorial for Espresso is very useful and comprehensive.

Define symbols:

- (+): correct input
- (-): incorrect input

Account Manager

Account Manager is a backend module, dealing with user accounts activity such as signing up, updating profile.

Sub-modules	Input	Output
Sign up/Registration	email, password twice	(+): redirect user to update profile (-): display error message
Sign in	email, password	(+): redirect user to main activity (-): display error message
Sign out	click <sign out> button	sign out
Update profile	new name, description, address	update profile, display success/error message
Reset password	<reset password> button	send email with code, enter code as password, direct to change my password

Product	Registration Module Unit Test
Date	02/07/2018

Author	Avinash Prabhakar		
Defect #	**Description**	**Severity**	**How Corrected**
1	When the user didn't provide either the password or the email, the app was crashing.	1	By checking the variables for null case we manage to protect the system from errors like that.
2	When the input to the password or username was too long the data sent to the database was crashing.	2	The length of the inputs were checked and if they are too long the user is warned to input a smaller input.
3	Email input incorrectly entered was crashing the system if not entered in proper format.	2	The email format is now checked to make sure it is something that can be sent to the database.

Product	Login Module Unit Test		
Date	02/07/2018		
Author	Altug Gemalmaz		
Defect #	**Description**	**Severity**	**How Corrected**
1	When the user info that is entered didn't exist in the database an error was generated crashing the app.	1	Now an error message is generated to let the user know that the input that is entered is incorrect. Utilizing FireBase built in systems this error is fixed.
2	When user tapped the login button twice very quickly the app was trying to send the information twice, consequently crashing the system.	3	By checking if the account existed on the database at the first tap we managed to ignore the second tap.
3	The back button on the UI wasn't working when tapped.	2	Appropriate variable updated to make sure that the back button could view the previous UI.
4	When the user didn't verify the email through the email	3	We utilized a FireBase API to make sure that the email is

	verification code, the app was still enabling access the user to his/her main view after log-in.		verified, after that check we enabled access to user to view his/her homeview.

Product	Edit Profile Module Unit Test		
Date	02/07/2018		
Author	Chaolun Shao		
Defect #	**Description**	**Severity**	**How Corrected**
1	The user can enter his or her name, address more than 50 characters.	3	Added a string checker to make sure the length of input is between 8 - 50 character.
2	The user can enter his or her description more than 250 characters.	3	Added a string checker to make sure the length of input is between 20 - 250 character
3	The user can enter same personal information when editing profile	3	Checked if the new information match the old information.
4	The user enter incorrect old password to update new password.	1	Checked if the entered old password match the correct old password.
5	The user should enter the same new password twice, but two password he or she entered is distinct.		Checked if the entered new passwords are same.

Product	Reset Password Module Unit Test		
Date	02/07/2018		
Author	Chau Minh Nguyen		
Defect #	**Description**	**Severity**	**How Corrected**

1	When the user email, linked with the user account, is entered incorrectly the app crashed.	1	The database is first checked to see it the user data with the email is existed, and then the process is continued.
2	When a non valid format email was entered the app crashed.	2	The email format was first checked to see if it was in invalid format. If that was the case then the user is displayed with a warning text.
3	Even though the user existed in the database sometimes the email wasn't received by the user.	1	Instead of our implementation we switched to a FireBase service which handled everything necessary for us once the email was received from the user. Through this FireBase is now guaranteeing sending email to the user.