Project Title: Password Protector
Team Members: Lei Liu, Christopher Gang, Christopher Sickler, Andrew Guilbeau
UT EIDs: ll28379, cg37877, cbs2468, abg926
Repository URL: https://github.com/chrisliu1234/461L-Project

Homework 3: Testing

EE461L Group 11 Password Protector

Unit Testing

For our project, we plan on doing a thorough unit testing on our project to check for bugs. There will be testing on both the front and back end. JUnit will be the focus of testing our project.

Standard JUnit tests and assertions will be used for the majority of our testing for the back end of our project (i.e. if the password is generated and stored correctly). For example, we will test to check that the password created meets all requirements that were chosen by the user and that the password was stored correctly onto the device. We will run many test inputs in order to check if our algorithm is error proof.

We will also do some minor UI testing for the front end of our application. We will test the functionality of all buttons and android activity pages. We will either test the application on our android devices or using the Expresso framework. This will ensure proper functionality of all buttons and activities.

Finally, we will test if our database is being edited properly. The application must be able to store and remove passwords so we will specify a test for this functionality. We will also ensure that only one password can exist for any account of a unique name in order to prevent possible user confusion.

Since we have an extensive understanding of our application's methods, our tests will consist mostly of white box testing. We want to achieve a full branch coverage when creating test cases in order to make sure every statement functions correctly. With full branch coverage, we will be able to test a majority of our application so a full path coverage approach will not be necessary.

Integration Testing

We will focus on UI testing for the integration testing segment. There are multiple transitions from one activity page to another, so all these transitions must be tested for correctness. We will test the following transitions:

- **LoginActivity <-> RegisterActivity**
    - If a new user tries to register then a transition to the RegisterActivity should be done. Once the new user registers, they will be returned to the LoginActivity page.
- **LoginActivity -> HomeActivity**

    - After logging in, the user will be transitioned to the home page to view, create, or delete passwords.
- **HomeActivity <-> CreateActivity**
    - If the user selects the option to create a new password from HomeActivity then the application should transfer the user to CreateActivity. After the user creates their new password, they will be returned to HomeActivity.
- **HomeActivity <-> ViewActivity**
    - If the user selects one of their current passwords to view, then the user should be transitioned from HomeActivity to ViewActivity to view their password information. Once the user is finished, they will be returned to the HomeActivity.

System Testing

For system testing, the internal workings are not critical to us so we will use a black box testing approach. We will focus system testing on checking if outputs are correct based on their corresponding inputs. We will use input space partitioning in order to cover most of the use cases.

The specific forms of testing that will be done include usability, installation, and exception handling. During system testing, we will check that the application is user friendly and can be successfully installed without any difficulty. In order to perform these tests, we will run the tests manually. We will also have other testers of our application in order to increase coverage of users and to receive more feedback on what to improve on.

Our group will also test for any exceptions from any external APIs if we decide to use any (i.e. GoogleSignIn). These tests will check that the use of these APIs will not interfere with our application's functionality.

Regression Testing

While developing our application, we will use regression testing in order to verify that our application still functions correctly when making edits to our existing code. After updating our code, we will run tests to see that the application still functions properly before making any commitments onto our git project. This will ensure a smooth development process without new code breaking our application's functionality.