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

Final Report
EE461L Group 11

Motivation

As most computer users have many accounts and might switch between computers. It is risky for a user to have one password for every account they own because if password theft occurs then all their accounts are compromised. It is also difficult to manage multiple unique passwords for many accounts. We are going to make an application that can create, delete and manage passwords for users so they can have a diverse set of passwords with minimal effort. This application will help users keep track of passwords and can provide it easily whenever a user needs it for a login. This application will also let users edit their passwords easily if they feel the need to update their old passwords. With this application, all users will have an application to manage all of their passwords for them and keep passwords diverse.

User Benefits

Users can keep reduce the redundancy of one password by keeping track of multiple, randomly-generated passwords. Password diversity can be achieved with our application with little effort. The application will also generate passwords for the user that will meet all user requirements.

Features and Requirements

Login: the user will register with a login and key

a. Login and logout functionality

Password List: the user can edit a list of their passwords

b. List will display password names for user to locate specific passwords easily
c. Passwords can be accessed for additional details
d. Passwords can be created, deleted and edited
e. Created passwords will accommodate user password criteria needs
f. Functionality will not require internet access

Non-functional Requirements: usability

g. Must be user friendly
h. No errors should occur while the application is operating
i. Easy installation
j. Low storage space requirement

Design

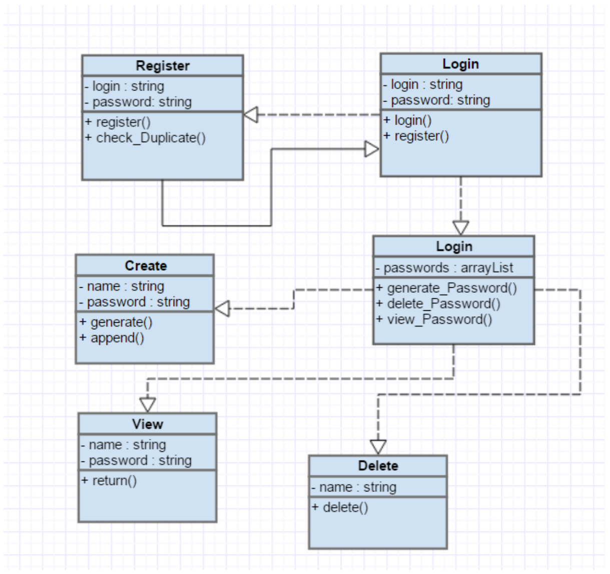


Figure 1: UML Class Diagram

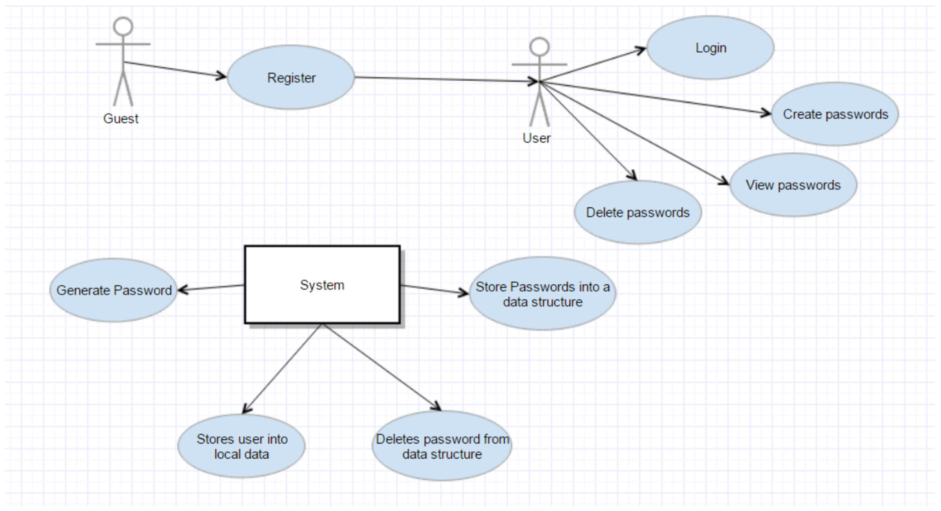

Guest
Register
User
Login
Create passwords
View passwords
Delete passwords
Generate Password
System
Store Passwords into a data structure
Stores user into local data
Deletes password from data structure

Figure 2: UML Use Case Diagram

No changes were necessary to our class diagrams. A guest will be able to register and create a key with the application and then login to either create, delete, view or edit passwords within the application. The system will do all the creating and storing for the user so none of the back end is relevant to the user.

Existing Libraries and Tools:

We used existing technologies to develop our project. For user interface design, we used mock-ups from Fluid UI. For version control, the team used Github to update and maintain our code in synchronization. Two tools we used to write our code include Eclipse and Android Studio. In Eclipse, we developed our back-end code along with all of our JUnit testing, and once we finalized our code, we injected our java code into our application in Android Studio. The team used SQL for database management.

User Interface

In PasswordProtector, there are 5 different activities, Main, Login, ViewPassword, NewPassword and Edit. The first activity that the user will encounter is the LoginActivity.

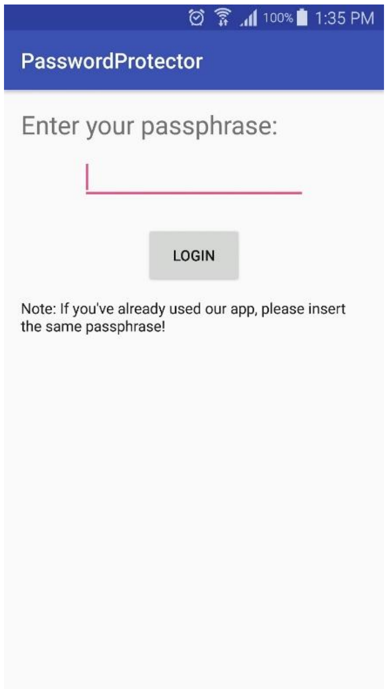
LoginActivity: The user opens the app and comes to this activity. There is a text entry location and a button on this page in the app. The text entry is for the user to enter their password for the app. This password is a passphrase the user creates on first use of the app. The user must remember this passphrase as this is the passphrase that is key to decrypting the database's encryption on the phone. Once the user inputs the passphrase into the textbox, they can press the button. If either there is no entry or the passphrase entry is incorrect, a message will appear telling the user that the passphrase entered is incorrect. However, if the passphrase is correct, the app will then continue to take you to the next page of the app, MainActivity.

MainActivity: Once unlocked, the app will display this page. On this page, there are 3 buttons and a selectable list. The list contains the names of the different accounts the user has created on this device. On first start-up of the app, the list will be empty as the user will not have created any accounts on this device. Each item in the list is selectable, and there are 2 different options the user may take once they have selected an account. The first is to delete the account, removing all the data from the database on the phone, making it completely unretrievable. The second is to open the account, which will then take you to the ViewPassword Activity. The last button is independent of the list item selected and is the button for creating a new account. This button takes you to the NewPassword Activity.

NewPasswordActivity: In this activity, there are initially shown 2 entry boxes for text and 4 different check boxes. The first textbox is for the user to enter the name of the account. The second is for the user to enter the length of the password they desire our password generator to make for their use, which is limited between 8 and 32 in length. The first 3 checkboxes are for the user to define what types of characters they want in their generated password. The three options are special characters, capital letters and numbers. The last checkbox is for the user to decide if they want to input any security answers for any security questions they may want to create for their account. When selected, this checkbox then makes 3 textboxes visible for the user to input their security answers. And lastly, there is a save button, which then creates the account defined by the user input and generates a password for the user to use, and then takes the user back to the MainActivity screen.

ViewPasswordActivity: In the MainActivity, an account was selected and the open button was pressed to take you to this activity. This activity is fairly simple as it shows the user the account name, password, and any security answers the user may have added. There are 2 buttons, one is a back button which will take you to the MainActivity, and the other is the edit button which will take you to the EditActivity.

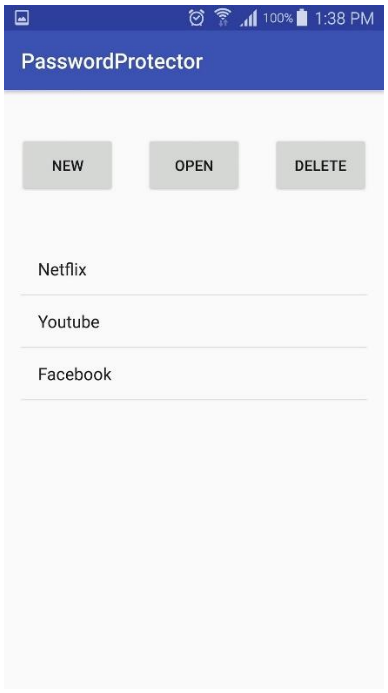
EditActivity: This activity is much like the NewPasswordActivity, except that in textboxes for the account name and security answers, the account information already made by the user is in there. The user may edit those as they please, and they may also generate a new password by inputting a length and making the selections of the different checkboxes with the parameters for the password generator. Once the user has done all they wish to do, they may select the save button which will then take them to back to the MainActivity.



Enter your passphrase:

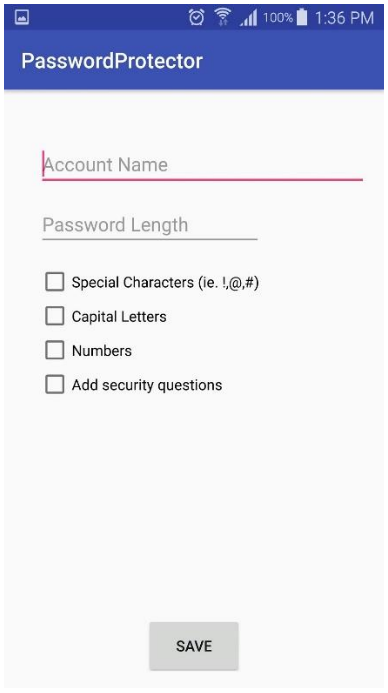
Note: If you've already used our app, please insert the same passphrase!

LoginActivity



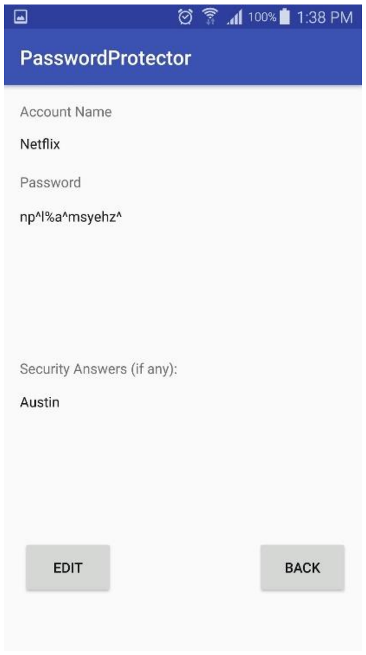
Netflix
Youtube
Facebook

MainActivity



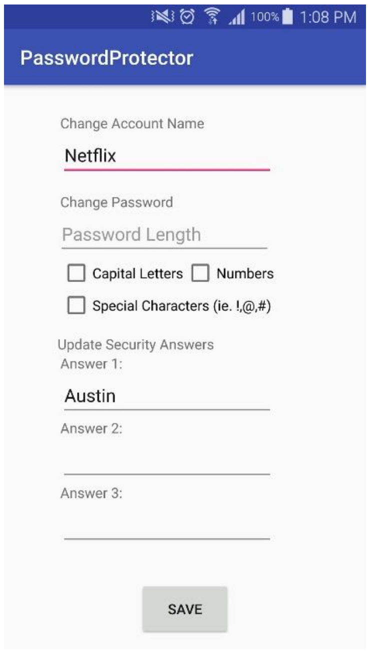
Account Name
Password Length
- [ ] Special Characters (ie. !,@,#)
- [ ] Capital Letters
- [ ] Numbers
- [ ] Add security questions

NewPasswordActivity



Account Name
Netflix
Password
np^l%a^msyehz^
Security Answers (if any):
Austin

ViewPasswordActivity



Change Account Name
Netflix
Change Password
Password Length
- [ ] Capital Letters
- [ ] Numbers
- [ ] Special Characters (ie. !,@,#)

Update Security Answers
Answer 1:
Austin
Answer 2:
Answer 3:

EditActivity

Testing
For our project, we did a thorough unit testing on our project to check for bugs. There was testing on both the front and back end. JUnit was the focus of testing our project.

Standard JUnit tests and assertions were 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 tested 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 ran many test inputs in order to check if our algorithm was error proof.

We also did some minor UI testing for the front end of our application. We tested the functionality of all buttons and android activity pages. We tested the application on our android devices. This ensured proper functionality of all buttons and activities.

Finally, we tested if our database was being edited properly. The application must be able to store and remove passwords so we specified a test for this functionality. We also ensured 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 consisted mostly of white box testing. We wanted to achieve a full branch coverage when creating test cases in order to make sure every statement functions correctly. With full branch coverage, we were able to test a majority of our application so a full path coverage approach will not be necessary.

Integration Testing
We focused on UI testing for the integration testing segment. There were multiple transitions from one activity page to another, so all these transitions were tested for correctness. We tested 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 used a black box testing approach. We focused system testing on checking if outputs were correct based on their corresponding inputs. We used input space partitioning in order to cover most of the use cases.
The specific forms of testing that were done include usability, installation, and exception handling. During system testing, we checked that the application is user friendly and can be successfully installed without any difficulty. In order to perform these tests, we ran the tests manually. We also had other testers of our application in order to increase coverage of users and to receive more feedback on what to improve on.

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

Regression Testing

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