Assignment 3 – Hybrid Android App for Chevereto

Hard Deadline: 23:59, April 20, 2018

TA-in-charge: Shuyue Hu

Introduction

In your second assignment, you used Chevereto to host your own image sharing website. However, given the popularity of mobile devices nowadays, a mobile app for your own website would help attract more potential users. There are two kinds of mobile apps: native and hybrid. The majority of the apps (e.g., Pokemon Go and Twitter) on your mobile device are native apps, which are written in languages that the platform accepts. They are fast and responsive, but also take a long time to build. To save the time and money, one good alternative is to build a hybrid mobile app. In this way, the functionality of your website, which is implemented in JavaScript and CSS etc., can be mostly reused and loaded in a WebView of your mobile app. With just a few changes, it takes you a shorter time to build a simple yet viable mobile app for your website.

In this assignment, your task is to build a hybrid mobile app for Chevereto. Specifically, your app should enable users with Android devices:

- to login to Chevereto;
- to view the content on Chevereto, while some adaptation is required to make the app more user-friendly;
- to upload images from their own devices.

Note that we aim for an easy task even for whom that do not have any prior experience on Android development. Therefore, many features which are necessary for a good image sharing mobile app are omitted to significantly downgrade the difficulty. You might see Instagram for an excellent example of image sharing hybrid app.

Implementation, Grading, and Constraints

- The minimum Android version should be set to Android 6.0.
- Implementing your APP with Java only.
- An ugly user interface is acceptable.
- You can use real Android devices (like Android phones and Android tablets) and any Android emulator to run or test your app. However, our grading environment is a virtual device (Nexus 6) provided by Android Studio.
- For submission:
  1. The server address should be set to http://10.0.2.2.

2. You should export your whole project and your apk (the debug version) to a single zip file named as "studentID.zip" and submit it through blackboard.

Task 1: Setting up Development Environment and Accessing Chevereto with Mobile Devices

Android Studio is the official IDE for Android development. You can download it at: https://developer.android.com/studio/index.html. After the installation, you can build your first application and try to run it to finish all the necessary configurations. There are two ways to test/run your applications: using your real Android devices and using Android emulators. Android Studio has its own emulator and provides several virtual Android devices. You can find and manage them via Tools -> Android -> AVD Manager.

To see how Chevereto looks like on a mobile device (which can be virtual or real), you can type the address of Chevereto via any web browser on your device. If you use a virtual Android device, please type: http://10.0.2.2 . Why not http://localhost ? This is because Android system considers itself to be the localhost. If you use a real Android device, you should connect your device to the same WiFi as that of your server, find the IP address of your server and type that IP address via your device's web browser. This should be what you will see in an Android browser:

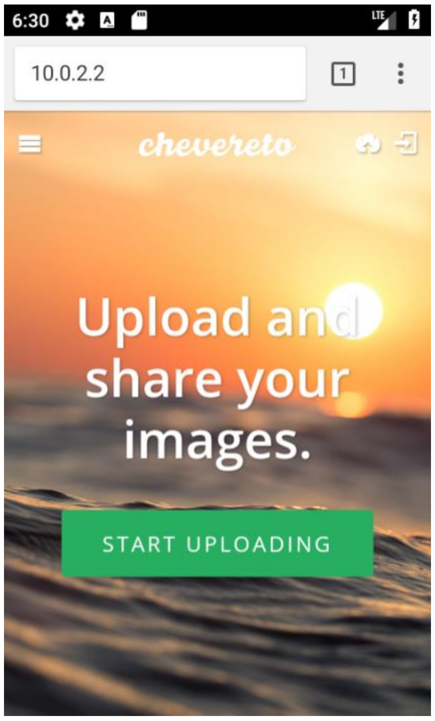


Figure 1: Chevereto on Mobile Devices

Have a try on Chevereto via web browsers of mobile devices! Think about an interesting question: what can be improved so as to provide better user experience? In the following tasks, you will do the most basic things, which are still far from enough, to achieve this goal.

Task 2: Handling Login

By default, to login to Chevereto on mobile devices, you have to first click the icon on the top right corner, and then type the email address and password in its login page which is shown in Figure 2. Obviously, this is not the usual way for mobile phone users to login.

Therefore, in this task, you first have to implement a login page for your app. An example login page is shown in Figure 3. Your login page does not have to look exactly the same. After a user has pressed the button "SIGN IN TO CHEVERETO", your app should send the data to server for authentication. To handle the networking and data transmission, traditionally, Android developers utilize either classes (i.e., httpClient and HttpURLConnection) provided by Android or open-source networking libraries (e.g., Volley, OKHttp3 and Retrofit2).

However, as you are developing a hybrid app, you can utilize Chevereto's original login page to simplify the data transmission. Indeed, your task is to send the user's credentials through Chevereto's original login page to complete the login process on behalf of the user. That is to say, although the user logins in your own login page, your app actually leverage Chevereto's original login page to talk to the Chevereto server and complete the login process. Specifically, your app should be able to get the user's email address and password, fill the data to the Chevereto's original login page, and programmatically click on the submission button to complete the login process. The procedure is shown in Figure 4. Note that this procedure should be hidden from the user. In other words, the user only directly interacts with your Login Activity, instead of the Login Page of Chevereto in a WebView.

After that, the user will either be redirected to the main page of Chevereto in a WebView (which will be implemented in Task 3) if the authentication is successful, or be reported with an error message in the same Login Activity if he/she fails to login. To downgrade the difficulty, let us just assume that all the users have already registered an account such that you do not need to implement a user account registration interface. However, they might forget their passwords or type the passwords incorrectly. Your app should be able to detect the login status and show the correct corresponding information to the user.

*Hint : You **might** accomplish this task in the following steps:*

1. *get the user's user name and password from your app's login page,*
2. *use WebView to load Chevereto's login page;*
3. *find the element of Chevereto's original login page, and set the element's values of email address and password to those of the user via injecting Javascript code into WebView;*

4. *submit the user credentials to Chevereto by finding and clicking the submission button of Chevereto's original login page via injecting Javascript code into WebView;*
5. *check whether the login is successful using JavaScript. Your code should be able to detect the authentication result. For example, a successful authentication request will result in Chevereto redirecting the user from the login page to the main page, while an unsuccessful authentication request will cause Chevereto to show some error messages. You can read the authentication code of Chevereto to find out how you can catch such events/data. Alternatively, you might also listen for navigation-related events. If you use the latter approach, you should double verify in the destination page if the user has successfully logged in.*
6. *call the JavaScriptInterface code exposed by Java in your JavaScript to notify the login result to your Login Activity; then start/switch to another activity that shows the main page/interface of Chevereto to the user if the login is successful.*

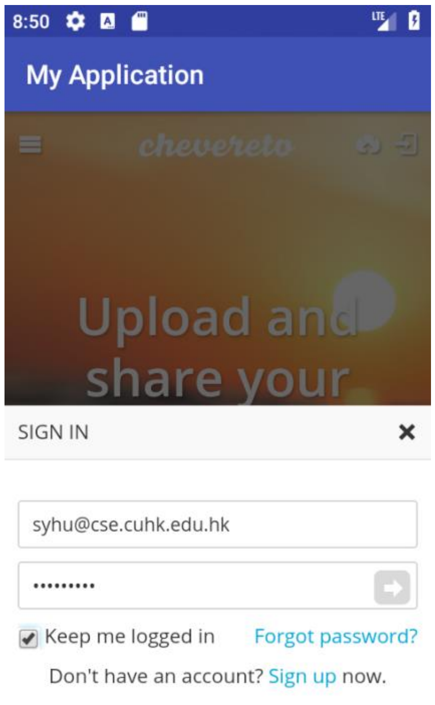


Figure 2: Original Login Page of Chevereto.

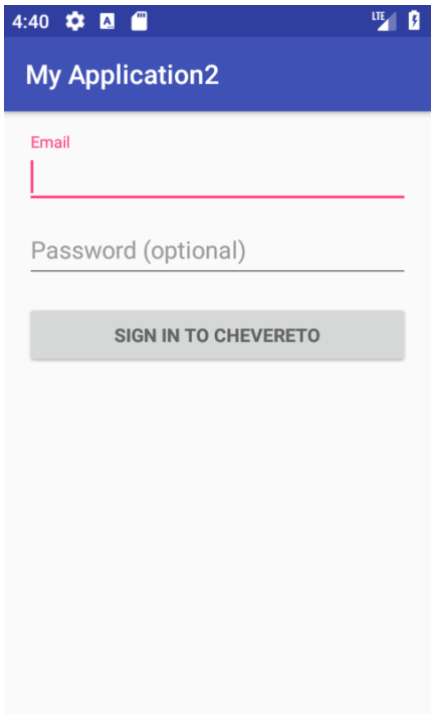


Figure 3: Your App's Login Page.

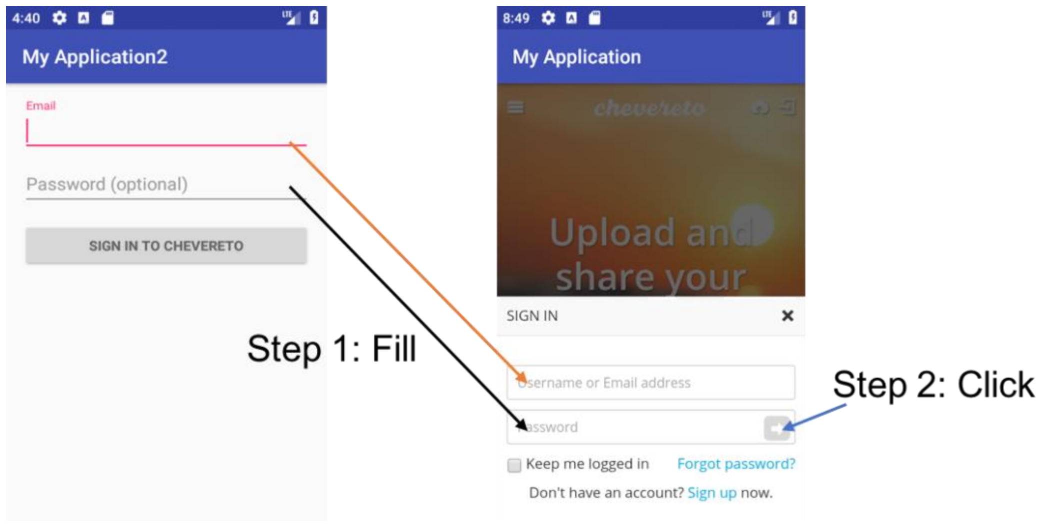


Figure 4: Procedure of faking a login operation in Chevereto's original login page.

Your app should also be able to persist the WebView cookies, so that the user is always in the state of login, when he/she is using your app to access Chevereto.

Task 3: Redesigning Interface

After the user has passed the authentication, your app should direct the user to the main page of Chevereto. Figure 5 is an example of what the user sees after successful login. As shown in Figure 5, some elements do not fit a mobile app so well. For example, there are two tiny icons/buttons (which have been circled in Figure 5) that support image uploading. Moreover, it is a common practice to place the commonly used buttons in the app bar.

In this task, your app should hide those buttons within the WebView and display them in the native application interface, i.e., the app bar, instead. An example is shown in Figure 6. When the user clicks the button "UPLOAD", the original Chevereto's image uploading page is displayed. Similarly, when the user clicks the button "MENU", the original Chevereto's menu page is displayed. Moreover, you should also beautify the original Chevereto's menu page by removing the button "UPLOAD" and moving the bottom two buttons upwards.

Hint : You might use ToolBar to serve as the app bar, and hide the buttons via injecting JavaScript code into WebView. Your app might set a listener for listening the click events of buttons "UPLOAD" and "MENU". Once the buttons are clicked, your app might automatically inject JavaScript code into WebView to display the pages.

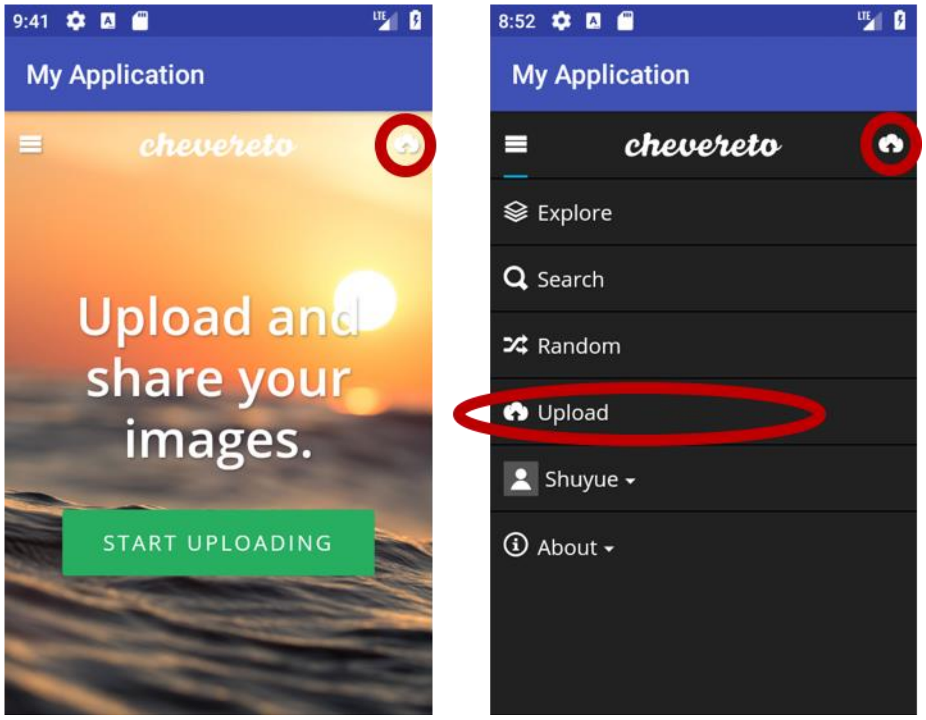
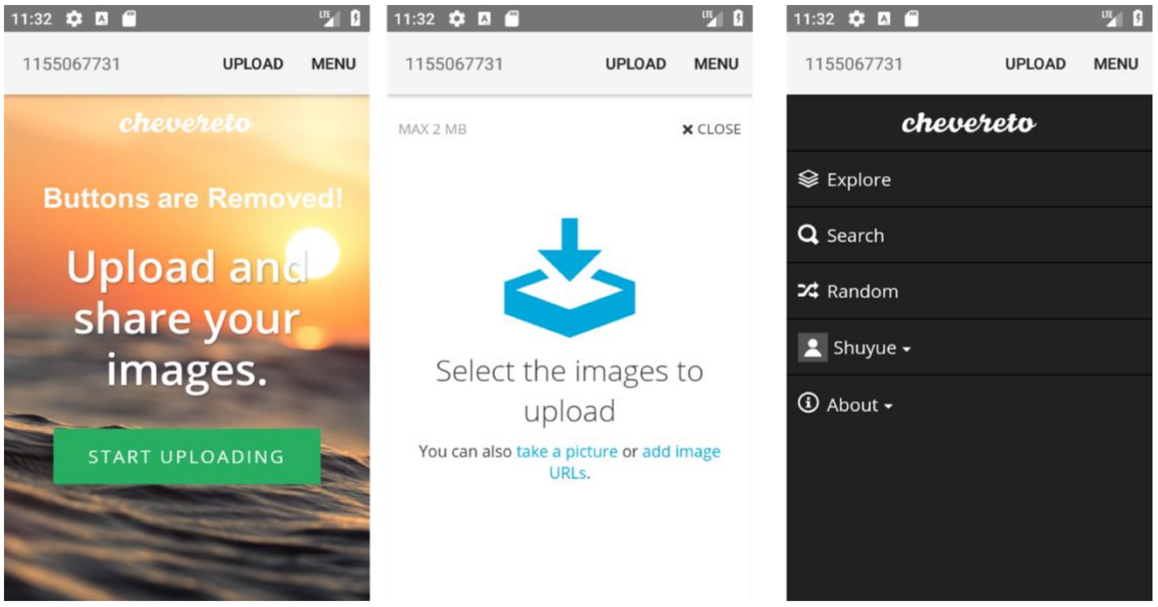


Figure 5: Original Main Page of Chevereto After Login.



(a) Buttons are not clicked. (b) Button “UPLOAD” is clicked. (c) Button “MENU” is clicked.

Figure 6: Target Main Page After Login.

Note that the default Android WebView does not allow file selection from the device. You need to provide such support such that the user is able to select and upload images from the device's photo library. Please do not use the AdvancedWebView class (https://github.com/delight-im/Android-AdvancedWebView) for completing this task.

When the user clicks a link from the web page in your current WebView, you should make sure that the user is able to view any Chevereto page within the current WebView instead of in another application.

Milestones

Task 2	**50%**
Login User Interface / Login Activity	10%
Implementing Login via Chevereto's Login Page	15%
Handling Login Results	20%
Cookie Management	5%
Task 3	**50%**
Hiding Elements in the Main Interface of Chevereto	6%
Redesigning the Menu Interface in Chevereto	6%
Adding Buttons to App Bar	6%
Implementing the Upload Button	11%
Supporting local file selection	10%
Implementing the Menu Button	11%
Total	**100%**