**Activity: Login**

The Launcher Activity listed in the AndroidManifest.xml. This is the first Activity that is loaded whenever the user starts the app. From here, the user can log in.

Initially the user’s login credentials were stored onto the phone. If the credentials existed, and the log-in token hadn’t expired, the user was automatically sent to the Main Activity. However, for security reasons, this has been removed, and the user is required to log in with their username and password every time they log in. The code still exists as comments, however, if there is a desire to allow the user to log in automatically, see if there is a way to retrieve the user’s encrypted password from the server first. Never store passwords on the phone, encrypted or otherwise.

**Manifest Details**

UserLogin.java is the class that implments the Login Activity. In the AndroidManifest, it has the noHistory parameter set to “true”, meaning UserLogin is not stored in the Activity stack. Pressing the Back Button while on the Main or Registration Activities will immediately close the application.

The screenOrientation of the activity is set to “portrait”

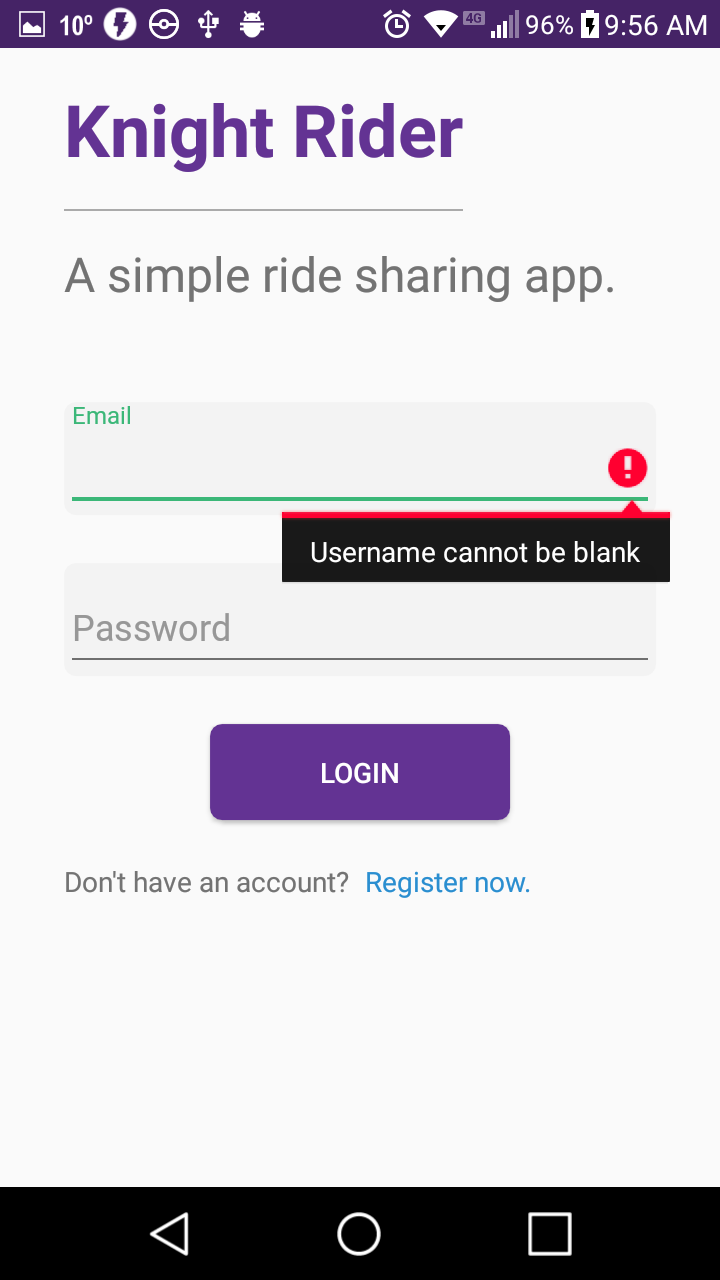
The Activity uses the theme”@style/AppTheme.NoActionBar”, which can be found in the res/values/styles.xml.

The Activity’s intent filter contains the activity’s name, and also indicate that this is the launcher activity.

**Layout File**

activity\_user\_login.xml

* The Activity’s sole layout file. It’s a basic interface with TextInputLayouts for entering the user’s email and password. It also has a clickable Login button that will attempt to validate and retrieve the user’s account. There is a clickable TextView that takes the user to the Registration Activty.
* The entire activity is contained in a ScrollView that holds a single ConstraintLayout. All visible objects are children of the ConstraintLayout.
* loginButton (Button):
  + onClick: login – calls the login(view) method within UserLogin.java to send a login request to the server.
* registerTextLink (TextView):
  + onClick: register – calls the register(view) method within UserLogin.java to open the Registration Activity



**Class File**

UserLogin.java implements the Login Activity’s functionality.

There are four private variables in the scope of the class:

* -usernameInput: TextInputEditText – Email
* -passwordInput: TextInputEditText – Password Text Field
* -prefs: SharedPreferences – Stores and retrieves the response token on the device
* -queue: RequestQueue – Holds JSON requests

The methods below will be explained in the order they appear at the time of writing:

* \*onCreate(savedInstance) : void
  + The method called with the Activity is first instantiated. It initializes the ‘queue’ and ‘prefs’ variables. It also presents the login’s layout, and initializes the username and password inputs to their corresponding Views.
  + Previously, this method also used ‘prefs’ to check whether the user’s login token has expired. If the token hadn’t expired, the Activity would attempt to sign in the user without displaying the login’s layout.
* -login(): void ---UNUSED---
  + This was the login method called from within the class to log in the user, using their stored credentials. This method has been commented out for security reasons.
  + It would have created a HashMap that store the username and password stored in the phone’s SharedPreferences. Next, it would have sent a JsonObject POST request to log the user into the server.
* +login(view): void
  + This login method is called from the layout.
  + After validating the user’s input, the user’s login credentials are temporarily stored in a HashMap, and sent in a JSONObject request to grant the user access to the app.
* -encryptPassword(password): String ---UNUSED---
  + Converted passwords into a SHA-256 string to be stored on the phone.
* -setTotkenExpirationDate(): void
  + When the user logged in, this would have taken the current time epoch, and added 7 days, marking the stored security token’s expiration date.
* -hasTokenExpired(): Boolean ---UNUSED---
  + Checked if the current time epoch had was greater than the expiration date’s epoch.
* +register(view): void
  + This method is called from the layout.
  + It simply starts the Registration Activity.