GitHub Repository Link: <https://github.com/bentemplin/brodents.git>

User Stories:

**Benjamin Yarmowich**

As a User, I want to login to the system, so that I can use the app.

Tasks:

\* Accept the User's inputted Username and Password

\* Search the database for the <key, value> pair of the Username and encrypted password hash

\* Grant or Deny access to app based on whether the pair is found

\* Offer a forgot password button to reset the password based on email authentication

Acceptance Scenarios:

\* Given there is a network connection, when the app is launched, app prompts log in.

\* Given the User enters a valid username and its corresponding password is in the database the main screen of the app is displayed

\* Given the User enters a valid username and its corresponding password is not in the database the User will be prompted with incorrect Username/Password combination

\* Given the User enters an invalid username the User will be prompted with incorrect Username/Password combination

\* Given the User enters 3 or more wrong username and password combination the User will be prompted to recover password and enter their email address or try again

\* Given the User enters 10 wrong username and password combination the User will be barred from entering a Username/Password for 1 hour

**Ben Templin**

As the App, I need to access the stored information, so that I can retrieve User and Report information.

Tasks:

\* Create a MySQL Database on a remote server

\* Configure remote login on the database

\* Write a custom Java class to connect a java app to the database

\* Write a subclass that has the login credentials for the Rat App Database

\* Create a .jar file with the custom Java classes and add it to the Android Project

Acceptance Scenarios:

\* Given there is a network connection, when the app is launched, it connects to the database.

\* Given the app is connected, when a valid database query is made, results are returned.

\* Given the app is connected, when a valid database update is made, the database is updated.

\* Given there are valid user inputs, when a query or update is made, the statement is parameterized to prevent SQL injection.

\* Given there is no network connection, when the app is launched, a message is displayed saying that the app can't connect to the database.

\* Given the app is connected, when an invalid database query is made, a message is displayed saying how to fix the error and the query is not made.

\* Given the app is connected, when an invalid database update is made, a message is displayed saying how to fix the error and the update is not executed.

**Rikesh Subedi**

As a User, I want to Register for an account, so that I can start reporting rats.

Tasks:

\* Implement a registration link on Log In page.

\* Create a registration form to gather user name, username, password, and location.

\* Query the database for an existing account with the given username.

\* Send credentials server-side if username not found.

\* Display notification showing success of account creation.

\* Redirect to main screen of the application.

Acceptance Scenarios:

\* Given the form is valid and complete, when the REGISTER button is pressed, an account is created in the database.

\* Given the form is not complete, when the REGISTER button is pressed, stay on the form and notify the user of missing information.

\* Given any state of the form, when the CANCEL button is pressed, return to the Log In page.

\* Given the user enters a username, when the user exits the field, text displays its validity.

\* Given the user enters an invalid username, when the form is submitted, stay on the form and notify the user of invalid information.

\* Given the user enters an invalid password, when the user exits the field, a reminder with password requirements will be displayed.

\* Given the user enters an invalid password, when the form is submitted, stay on the form and notify the user of password requirements.

**Austin Ayers**

As a User, I want to logout of the system, so that someone else can use the app

Tasks:

-Accept a button press to log out of the app

-Return the log in screen after a successful log out

Acceptance Scenarios:

-Given that the user is logged in, when the user presses the logout button, then the app logs the user out

-Given that the user successfully logs out, when the user returns to the screen, then the app displays a new log in screen