Michael Kozak (Student ID: 10223568)

**Project:** Credibility Assessment Android App

**Purpose:** Give the user the ability to analyze their contacts and generate a “Credibility Score” for them. This is based on theory known as “Type-Token Ratio” which states that the ratio of unique words to total words is an indicator of honesty.

**Skills Learned (the following aspects of the assignment were learned for this project):**

* Android Development (Activities, Interacting with OS, user permissions, UI elements, building/deploying, running in a VM, etc.)
* Android Studio and associated Eclipse integration
* XML-based UI development (previous experience was with Java Swing)
* Gradle Project Building
* JSON parsing
* GIT on Windows

**Source Code**: <https://github.com/mikewkozak/CS575_assn3>

**Video Walkthrough**:

**Requirements:**

* Android Studio and/or Eclipse
* Android SDK for Android version 23

**Known issues:**

When running in “Live” mode (See: To Configure), the application will load and parse text messages but it will not be able to associate those messages with the names in the spinner. This is due to my inability to test the program on my android VM as it has not sent or received any text messages.

**To Build:**

The system can be imported either in Eclipse or Android Studio (which uses the eclipse project file to generate a new Studio project and create the required Gradle files). Once imported, it should build automatically

**To Run:**

In either Eclipse or Android Studio, once the project is imported, simply choose “run” to launch the VM. In Eclipse, use the default “Android Application” run configuration if it does not import from the project file.

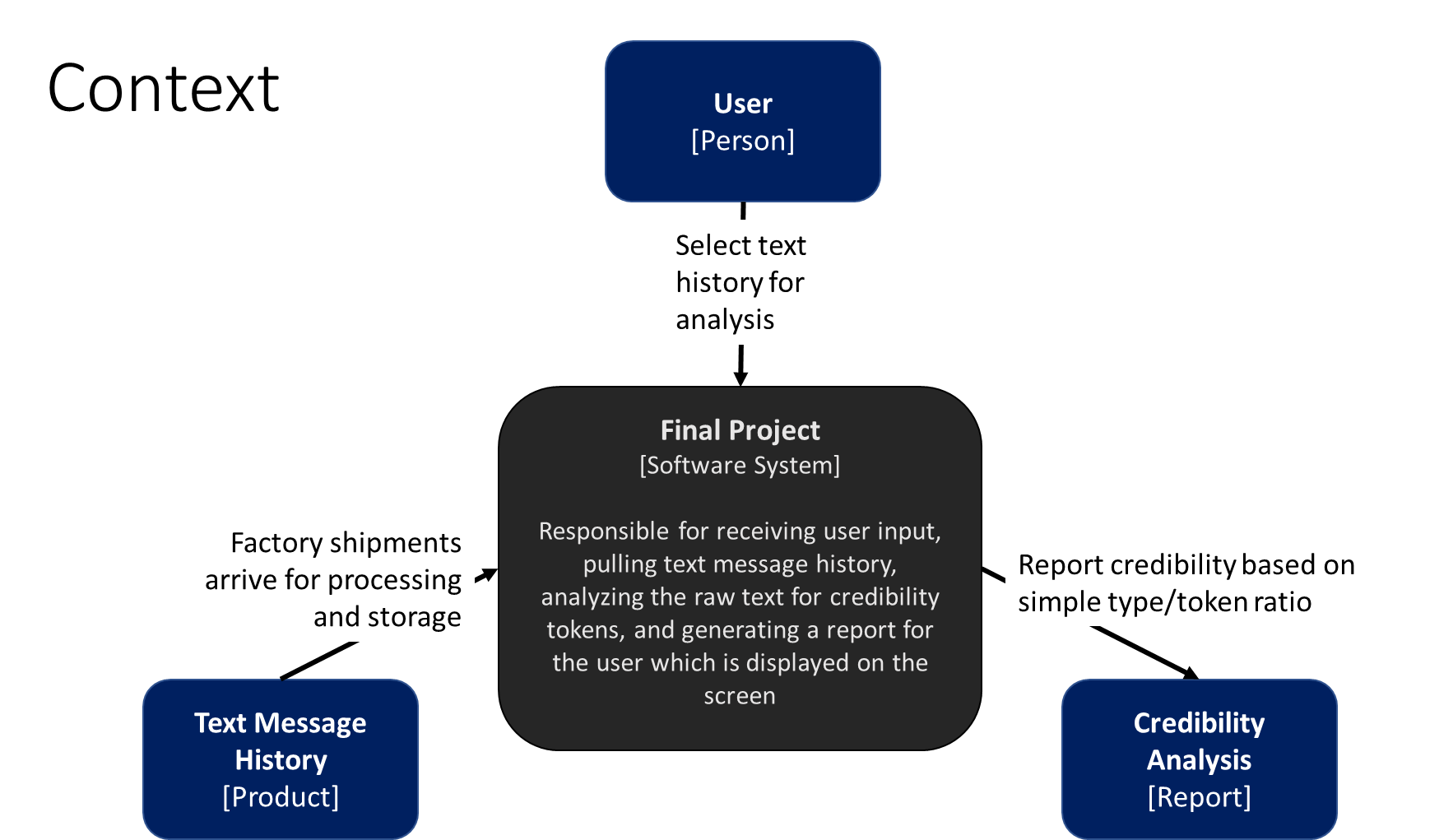
For the more adventurous, the APK generated can be installed to a real Android device with the correct OS version and run like any other program.

**To Configure:**

In MainActivity.java there is a *USE\_MOCK\_DB* constant at the top of the class. It is set to true by default and loads all the test data from a JSON file. If this is set to false (“live” mode), it will attempt to load any contacts and text messages from the Android OS it is running on by requesting permissions for that data and performing actual queries. For a VM this information is likely empty, but it should properly pull that data from any real device the APK is installed to.

**To Test:**

Feel free to enter text into the raw text field or select a user from the drop down. In both instances, the TTR, or Type-Token-Ratio for the text or contact’s text messages will be generated and displayed on the lower half of the screen.

**Architecture Explanation**:

The architecture of this project, while relatively small, attempts to embody a number of principles taught in the class. At the top level, the system takes user input as well as system data in the form of contacts and text messages, and produces credibility analysis reports for the user to digest.

Beyond this simple analysis, the system attempts to capture a number of high level architectural design decisions. The UI follows a MVC style with the View being handled almost entirely by Android. User interactions are captured by the main Activity which manipulates the datamodel to produce new info for the view. Requests for reports are sent out to adapters built on top of a common interface which interacts with either the Android OS or a mock DB loaded from JSON to return the data required. This data is then pushed into a report generator designed to pipe the request through multiple filters to generate the final report that the controller passes to the view. The code itself is segmented into a number of packages and as by passing in context where necessary, no class needs to know anything about the state of other classes. To avoid nulls in report generation an “Empty” version of the TextMessage class was created and is used when no results are found. Finally, a util class captures all the static functions required to analyze the text in a way that embodies good functional programming practices. Finally, although not explicitly required, code is formatted to be readable, with lots of inline comments, spacing, and proper use of variable names as to make maintenance less of a hassle.

