Design Document

For the Photo Tracker Application

11/20/2013

Code Ninjas

Eileen Balci

Priyaranjan Parida

Table of Contents

Table of Figures

1 Application Concept

The Photo Tracker Android application will be a tool that allows a user to take photos on their Android device and the pictures by location on a map. Users will be able to track places they have visited and taken pictures, they will be able to view the pictures they took at the location and be able to track the date they took the picture at the location.

1.1 Purpose

The purpose of this document is to provide an overview of the project design. This document includes the methodology used to manage the project as well as the design concepts, diagrams, mock-ups etc. to be used to develop a functional prototype of the Photo Tracker application.

1.2 Methodology

The software will be developed using an iterative rapid prototyping approach commonly referred to as the Agile Engineering development life cycle. The team will only have 3 weeks to develop a functional prototype of the application by the time development starts. To accomplish this the following tasking plan has been created.

- Week 1: Develop back end services and prepare hooks to tie into Android application based on documentation outlined in this design document
- Week 2: Develop the Android application with all user facing views completed (user interfaces) referencing the design outlined in this design document
- Week 3: Add functionality to Android views and connections to backend services/servers as stated in this design document

By the end of week three, the application shall be a functional application that allows a user to register, sign in, and view a map. This will be the basic functionality provided for the first version's, first iteration prototype of the application.

Additional iterations would take place to evolve the application until the functionality is fulfilled. During each iteration testing will be a constant that is done by the developer to ensure the code is working as expected for the scope of that iteration. Regression testing will also take place during each iteration to ensure that newly introduced code has not interfered with previously working functionality.

A description of future work and features can be found in Section 4 of this document.

1.2.1 Task Tracking

A task-tracking tool called Trello[1] will be used by the team members to keep organized and on target to meet the final deadline. Trello is a simplistic tool that allows a team to create columns, for example "To Do", "Doing", "Done" and insert tasks in the appropriate column. It helps give a visual representation of what needs to be accomplished vs. what is being accomplished or has already been accomplished to ensure the project is on track.

[1] Trello is a free online task tracking tool http://trello.com/

Figure 1 is a screenshot of some of the initial tasking added to Trello for this project.

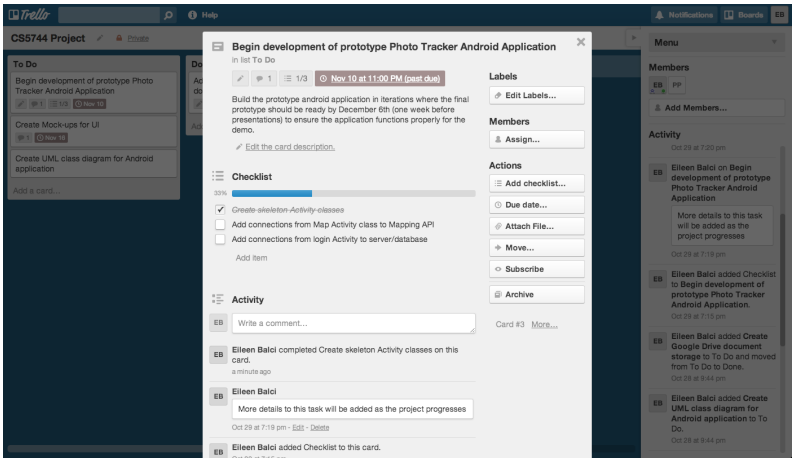

Figure 1 Screenshot of Trello

1.2.2 Version Control

Version control is very important for all development projects whether there are one or multiple developers on the project. In order to ensure developers aren't overwriting other developers code during the development phase, version control tools become quite handy. For this project the team will be using Git for version control. A Git Repository[2] has been set up to maintain the code base for version 1 of the Photo Tracker application.

Figure 2 is a screenshot of the GitHub repository created for the Photo Tracker application.

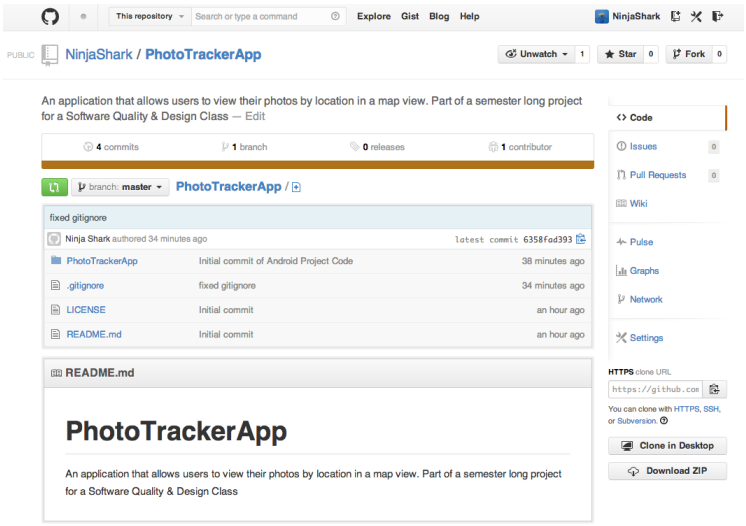

Figure 2 Screenshot of the GitHub Page

[2] The GIT repository can be found here: https://github.com/NinjaShark/PhotoTrackerApp

1.2.3 Diagram Creation

Various diagrams were created for the design document such as UML class diagrams, sequence diagrams and an activity diagram. These diagrams were created using an Online Diagram Creation tool called Creately[3]. A screenshot of what the tool looks like can be seen in Figure 3.

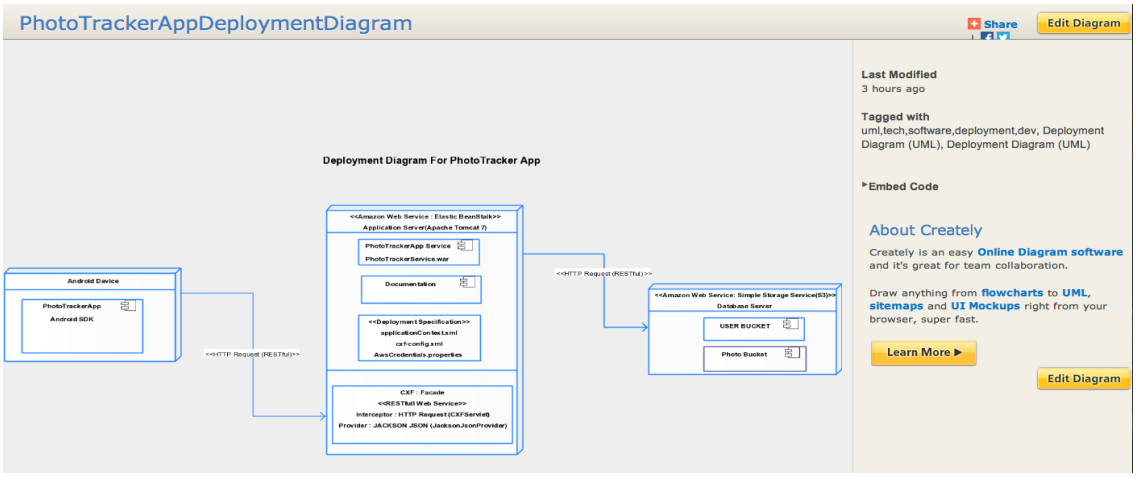

Figure 3 Screenshot of the Creately Tool

1.2.4 Documentation

The prototype application will contain documentation within the code using Java Doc and inline comments. This documentation is essentially part of the code base and thus is managed in the Git repository. The Design document, mock-ups, and diagrams will be stored on a Google Drive accessible by the team members to allow for centralized document storage and multi user editing capability on certain documents.

1.2.5 Testing

The team will continuously test code for the application and backend services as the prototype is built. This will ensure that the prototype being built will be functional when completed and have minimal bugs left to fix. Continuous regression testing will also be performed to ensure that new code did not create a bug in functionality that previously worked.

Best practices indicate that unit testing or testing suites for automated testing are the best way to ensure all cases are tested. Because of the minimal amount of development and testing time, unit tests will not be used for the first version's, first iteration of the application.

[3] Creately can be found here: http://creately.com/

1.3 Scope

The Photo Tracker application, version 1, will be focused on providing the initial basic capability to the user. Version 1 will allow a user to register and sign in to the application, take pictures, toggle location service preferences and view the pictures saved to the device on a map by location. For pictures that do not have location data (because location services were disabled) will not be shown on the map since they have no location data associated with them.

Iteration 1 for this version will include the following functionality:

- User shall be able to Register, this includes full backend service functionality
- User shall be able to Login, this includes full backend service authentication
- User shall be able to view a Map, using the MapQuest API

It can be tempting to add additional functionality to an application during development that is not part of the original requirements list. These features must instead be written down and be considered for future iterations or versions of the application. Feature creep can derail development timelines and cause additional issues in the program thus prolonging development time and cost.

2 Requirements

The Photo Tracking application will adhere to the following requirements that fall within the scope of the project.

2.1 Functional Requirements

The following requirements are functional requirements for the Photo Tracker Application version 1.

- The application shall provide the end user the ability to login with a username or password
- The application shall provide the end user a way to create an account if they do not have one already
- The application shall send account authorization data to the server to check validity against what is stored in the database and authenticate the user (or not authenticate the user if not valid)
- The application shall provide the end user the means for taking a picture
- The application shall provide the end user the means to view a picture
- The application shall provide the end user the means to view details of a selected picture (date taken, time, etc.)
- The application shall provide the end user the means to view a pictures location on a map

2.2 Non-Functional Requirements

The following requirements are non-functional requirements for the Photo Tracker Application. Non-functional requirements are requirements that depend on things such as hardware, processing power, performance, and other such metrics. These requirements may affect or be affected by functional requirements, but they are not functional requirements themselves.

- The application shall be able to support a TBD number of concurrent users
- The application shall be able to store a TBD amount of data per user

3 Overall System Design

This section encompasses the overall system design to create the Photo Tracker application prototype. In short, the system will use backend services, access to a mapping API, a way for the Android application to communicate to the backend services and providing the user an intuitive user interface to use the capabilities provided by the overall system.

3.1 Backend Services

Backend services will be required to handle user registration, authentication and data storage. While the first version for this application could be done locally on the device, the Photo Tracker

application has future version features like the ability to share photos and maps with friends. This would require an account creation process and storage of users and user data externally. Because of this known future feature, the team decided it best to start out with a user registration and login that is done externally to help ease the process of implementing this future feature when the next version(s) are developed. That future feature helped drive some of the architectural and design decisions for this application.

3.1.1 Services and Frameworks

The Photo Tracker App Server side will be built using the Spring Framework[4] along with the Apache CXF framework[5]. The Spring Framework is also supported on Android[6] and will be able to provide connection between the Android application and Backend services.

On the Data Access Object (DAO) layer an Amazon Web Service (AWS)[7] and Simple Storage Service (S3)[8] will be used to store User information and photo information with the geo location data in the cloud. The Photo Tracker Service will provide Software-as-a-Service (SaaS) where multiple RESTful Web Services[9] will be exposed for the android app (devices) to consume.

Services Beans will be managed by Spring Configuration. The Spring framework will take care of the post construction and pre destruction necessities along with the life cycle of Service beans. CXF Interceptors will handle all HTTP Request with a payload and use the JACKSON JSON[10] service providers to serialize and de-serialize the payload objects.

The Photo Tracker Service shall be build using the Eclipse IDE with the Maven plugin[11]. Maven will be used for dependency management and for the building and deployment process. The PhotoTracker AppService will be built with the specified Project Object Model (POM) file and an executable Web Archive (WAR) file will be generated for deployment.

3.1.2 Database

The Photo Tracker Service DAO layer is going to be built with Amazon S3 and will stores user data and photos in a secure and easily accessible cloud. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable,

---

[4] Information about the Spring Framework can be found at http://projects.spring.io/spring-framework/
[5] CXF is a combination of Celtix and XFire, thus the acronym CFX. More information about the Apache
[5] CXF Framework can be found at http://cxf.apache.org/
[6] Additonal information about Spring for Android can be found at http://projects.spring.io/spring-android/
[7] More information about Amazon Web Services can be found through http://aws.amazon.com/
[8] Information about the Amazon S3 can be found at http://aws.amazon.com/s3/
[9] More information about RESTful Web Services can be found at
[9] http://www.ibm.com/developerworks/webservices/library/ws-restful/
[10] More information about the JACKSON JSON can be found at https://github.com/FasterXML/jackson
[11] More information about the Maven plugin can be found at http://maven.apache.org/eclipse-plugin.html

secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites.

Each object is stored in a bucket and retrieved via a unique, developer-assigned key. A bucket can be stored in one of several Regions. A developer can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users. It also uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

3.1.3 Server

The Photo Tracker Service is deployed with the Amazon Elastic Beanstalk and runs the web application in a Tomcat 7 cluster consisting of 1-4 nodes in a secure and readily available cloud. AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with popular programming languages such as Java. A developer can simply upload their application and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling and application health monitoring. At the same time, with Elastic Beanstalk, the developer can retain full control over the AWS resources powering their application and can access the underlying resources at any time which is the Amazon S3 in the Photo Tracker application's case.

The endpoint of the deployed Photo Tracker Service in the Amazon Cloud with Elastic Beanstalk[12] can be seen in Figure 4.

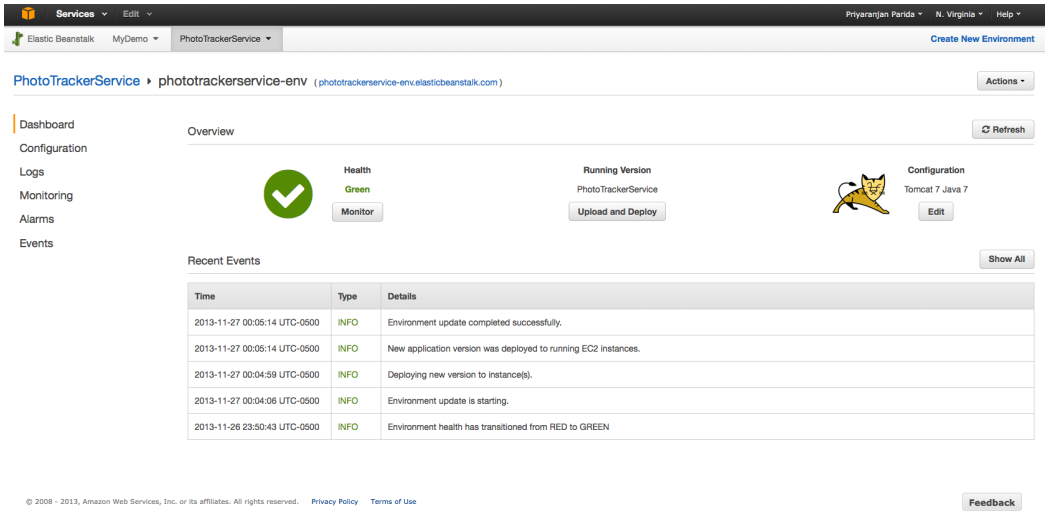

Figure 4 Screenshot of the Photo Tracker Service in the Amazon Cloud with Elastic Beanstalk

[12] Access the Photo tracker Service here: http://phototrackerservice-env.elasticbeanstalk.com/

3.1.4 Mapping API

The Code Ninjas used a MapQuest Mapping API[13] for Android. This API allows for developers to display maps and overlays on maps on an Android device. The map is by far the most predominate feature the Photo Tracker application provides. It is where the user will see pins marking the locations of where pictures were taken and will be able to view basic information about those photos when a pin is selected.

3.2 Android Application

The Android application is what the user will be interacting with. It is the interface for the user to be able to register, login, take pictures, view pictures and view the map.

3.2.1 Android Application User Interface Mock-Ups

Before development of the application started, some mock-ups of the user interface were created. The initial mock-ups were hand drawn and are shown below in Figure 5.

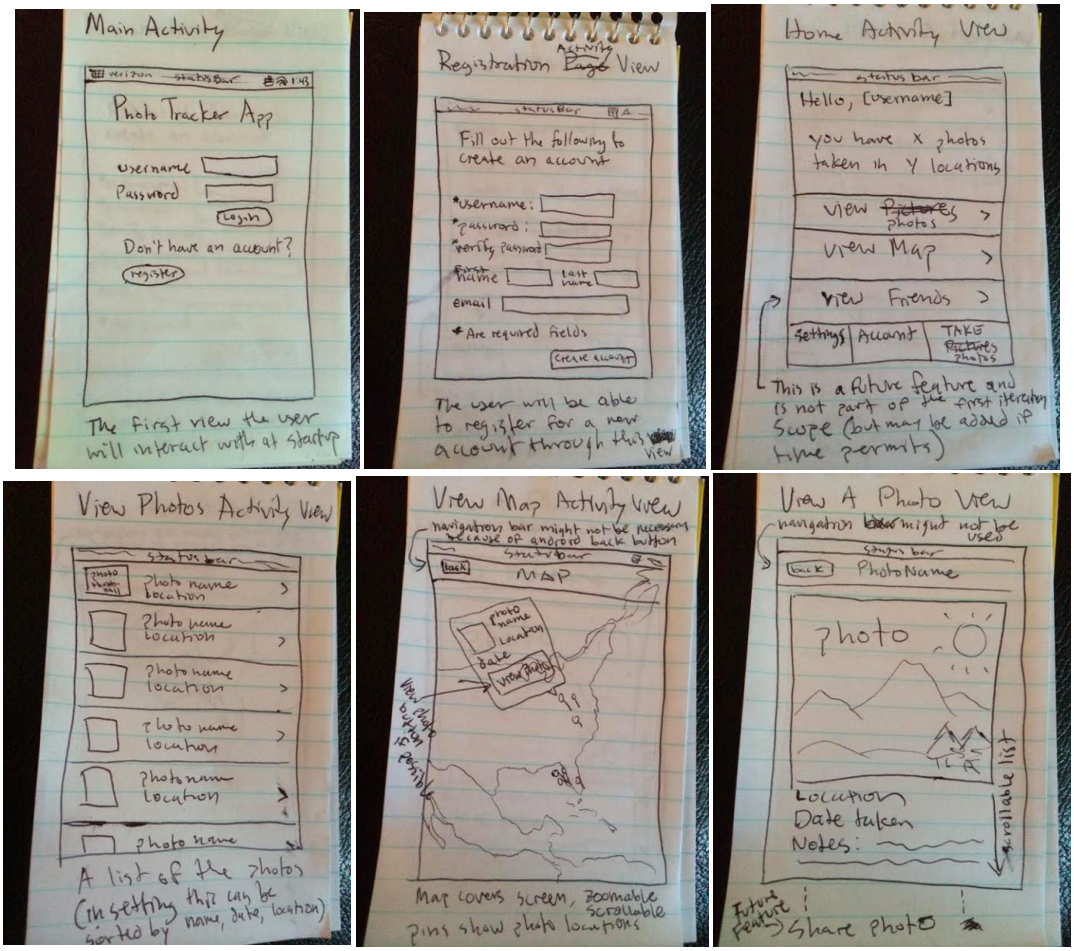

Figure 5 A Collection of Hand Drawn Mockups

[13] Mapping API: http://developer.mapquest.com/web/products/featured/android-maps-api/documentation

Before developing the functional code each Activity view's user interface was designed first using Androids user interface builder that is accessible through Eclipse. Originally a program called Balsamiq[14] was going to be used to design the mockup views, however, the license that the team had access too apparently expired. To avoid paying $79 for a new license, the team instead utilized Androids built in Interface Builder that is added to Eclipse when the Android Development Toolkit (ADT) plugin is installed. This interface builder allows developers to drag and drop components such as buttons, text boxes and so-on onto a layout representing the device. The components can be moved around later, as development progress, but makes for an easy visual for the developer as minimal code needs to be written to add the actual user interface components.

Once the components are added, the developer can add functionality to the component and change the components appearance through code. The following screenshots in Figure 6 are the initial Activity views based on the initially hand-drawn mock-ups. Some of the views may vary slightly from the hand-drawn versions to help improve upon the original concept.

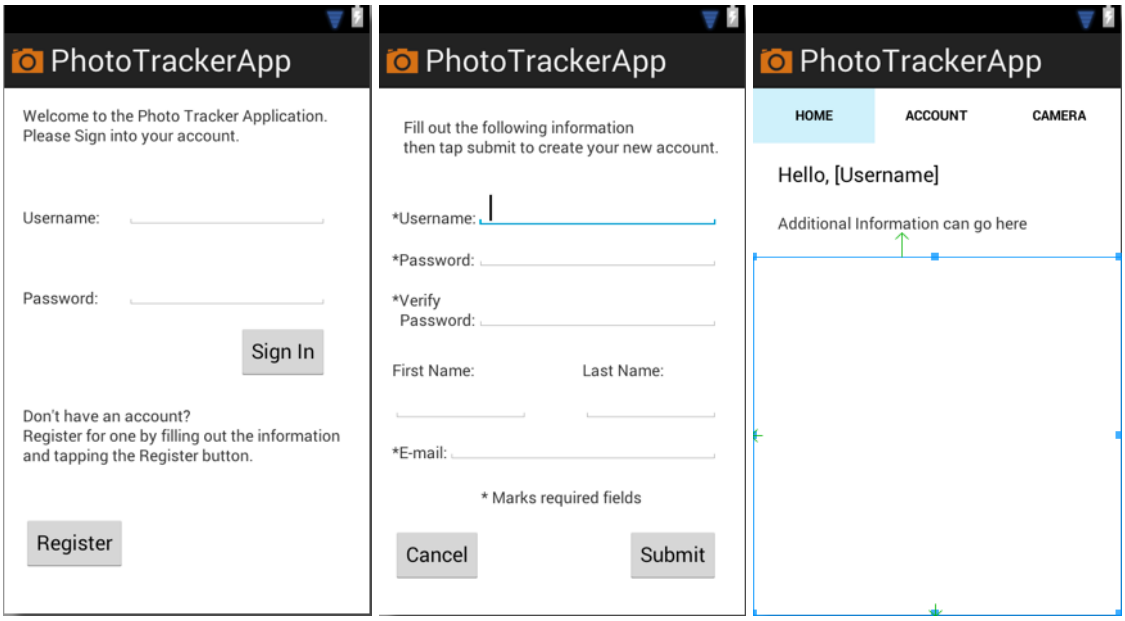

Figure 6 A Collection of Screenshots from the Interface Builder Mockups

The plus side to using the Interface Builder is that it can be applied directly to the development implementation when completed. Once the views were mocked up, all that is needed is Activity classes to be created that reference the various layouts and components in those layouts that make up the view.

[14] Balsamiq Website: http://balsamiq.com/products/mockups/

3.2.2 Getting Geo Location from an Image

Android photos are taken and saved as JPEG. When location services are enabled, the image will retain latitude and longitude data within the JPEG. This data is part of the Extensible Image File (EXIF) format[15]. EXIF data can be retrieved from images and used later. For this application the EXIF data for the latitude and longitude would be retrieved and stored locally for each picture so that the picture can be viewed on the map as a pinned location.

3.2.3 User Input Validation and Verification

The application will allow users to register for an account and be able to login once registered. Anytime users are given the chance to input data, the data should be validated and sanitized. This is to help prevent things such as SQL injection attacks but also help prevent runtime errors from occurring due to mismatched data inputs.

The registration view will have various text fields where the user will input data. This includes a Username, Password, Password Verification, First Name, Last Name and Email. For this prototype the following constraints shall be met:

- **Username** must be alphanumeric with the addition of allowing underscores
- **Password** must be at least 8 characters long, have at least 1 uppercase letter, at least 1 lowercase letter and at last 1 number or special character
- **Verify Password** must match the Password provided by the user
- **E-mail** must meet the standard email format such as username@domain.com
- **First Name and Last Name** are not required fields so they can be empty

After the user registers, their information will get saved in a database on the server. Now the user can sign in with the credentials they created. The login view shall have two text fields for the user to input their username and password. The fields should hold the same criteria as the corresponding fields have in the registration view.

When the user taps the “Sign In” button the username and password entered is checked in the database. If the combination matches, the user becomes authenticated and can start to take pictures, view pictures and view pictures on a map.

[15] More about EXIF can be found at http://exifdata.com/

3.3 Lightweight Use Cases

The following is a lightweight use cases for the Photo Tracker application registration and log in/out followed by a use case diagram seen in Figure 7.

Registration And Log in/out

Brief Description

This use case describes how a user registers and logs in/out the Photo Location Tracker System.

Basic Flow

This use case starts when an actor wishes to log in/out the Photo Location Tracker System.

1. The system requests that the actor complete the registration form with username, password and email.
2. The actor enters username, password and email in the registration form.
3. The system creates the actor (user) account and saves the actor details.
4. The system requests that the actor enter username and password.
5. The actor enters username and password.
6. The system validates the entered name/password and logs the actor into the system.
7. The actor clicks the logout button.
8. The system saves user information and terminates the session.

Alternative Flows

Invalid Name / Password

If in the *Basic Flow* the actor enters an invalid name and/or password, the system displays an error message. The actor can choose to either return to the beginning of the *Basic Flow* or cancel the login, at which point the use case ends.

Pre-Conditions

None

Post-Conditions

If the use case was successful, the actor is now logged into the system. If not the system state is unchanged.

Use Case Diagram

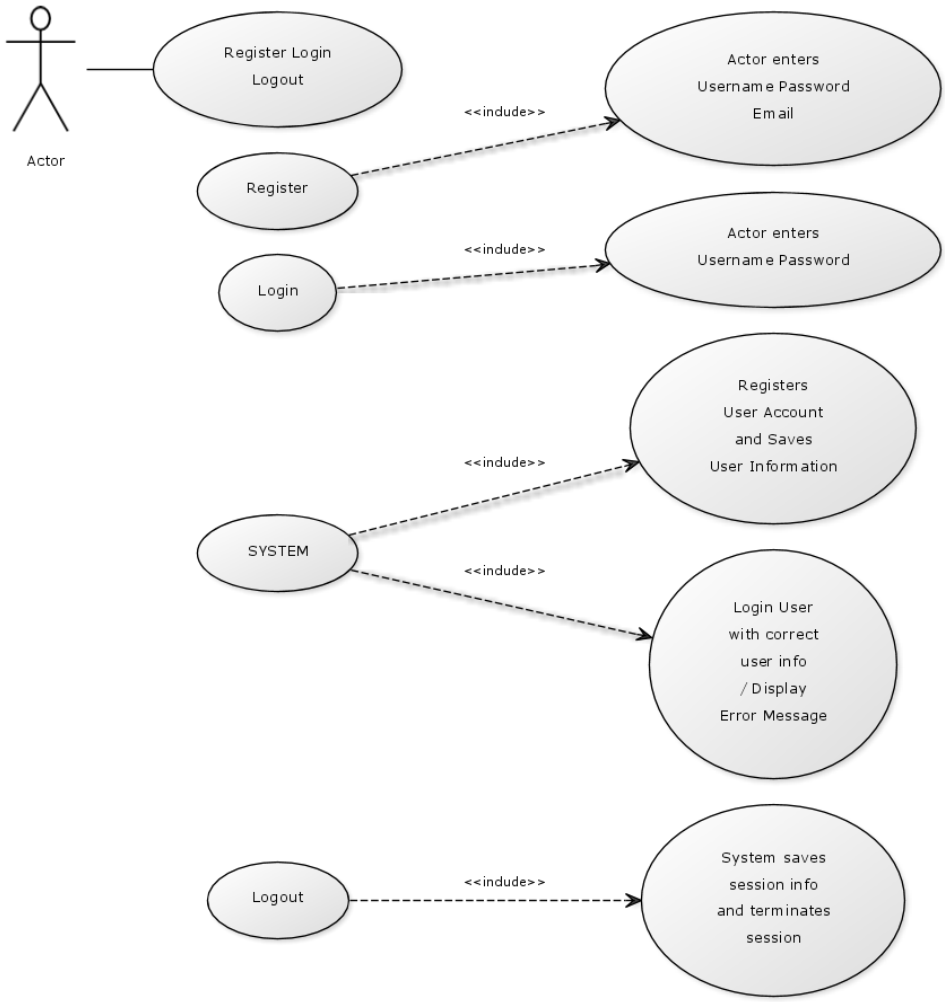

Figure 7 Use Case Diagram for Registering, Login and Logout

3.4 Sequence Diagram

The following sequence diagrams represent the Registration and Login functionality for the Photo Tracker application. The diagrams can be seen in Figure 8 and Figure 9.

REGISTRATION

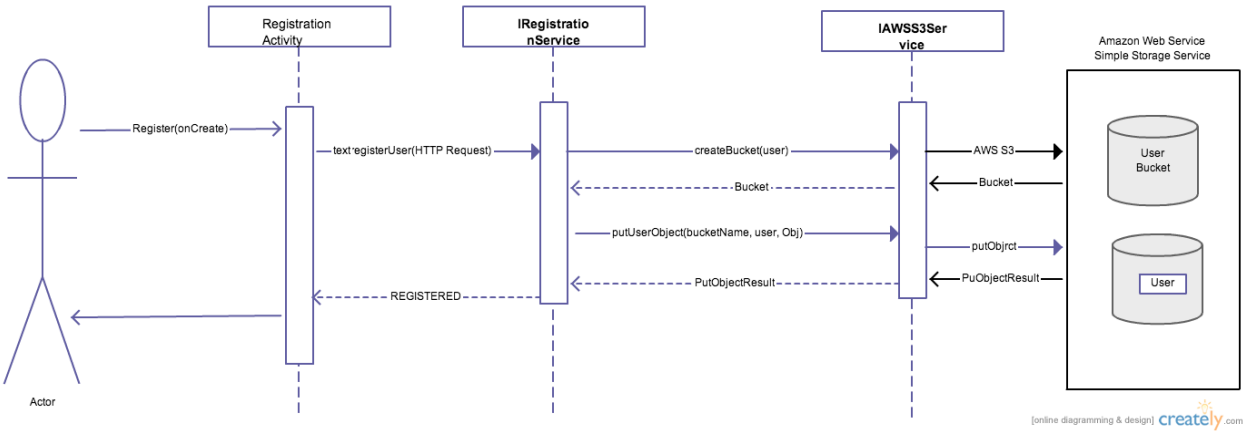

Figure 8 Registration Sequence Diagram

LOGIN

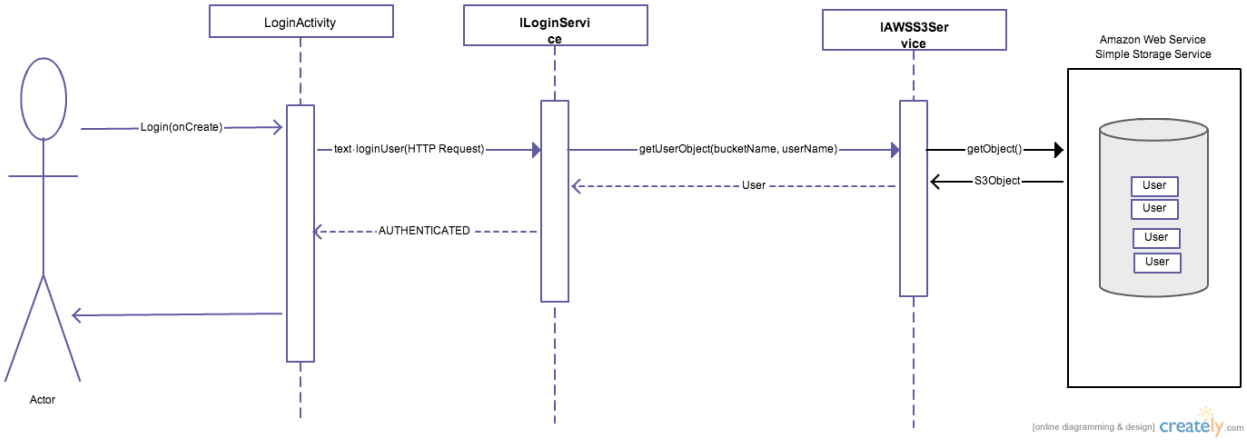

Figure 9 Login Sequence Diagram

3.5 UML Diagram

The following UML Diagram seen in Figure 10 outlines the Registration, Login and Mapping Activities and the utility classes used in the Android application. It also shows the services on the backend server and how they are connected to the Android application.

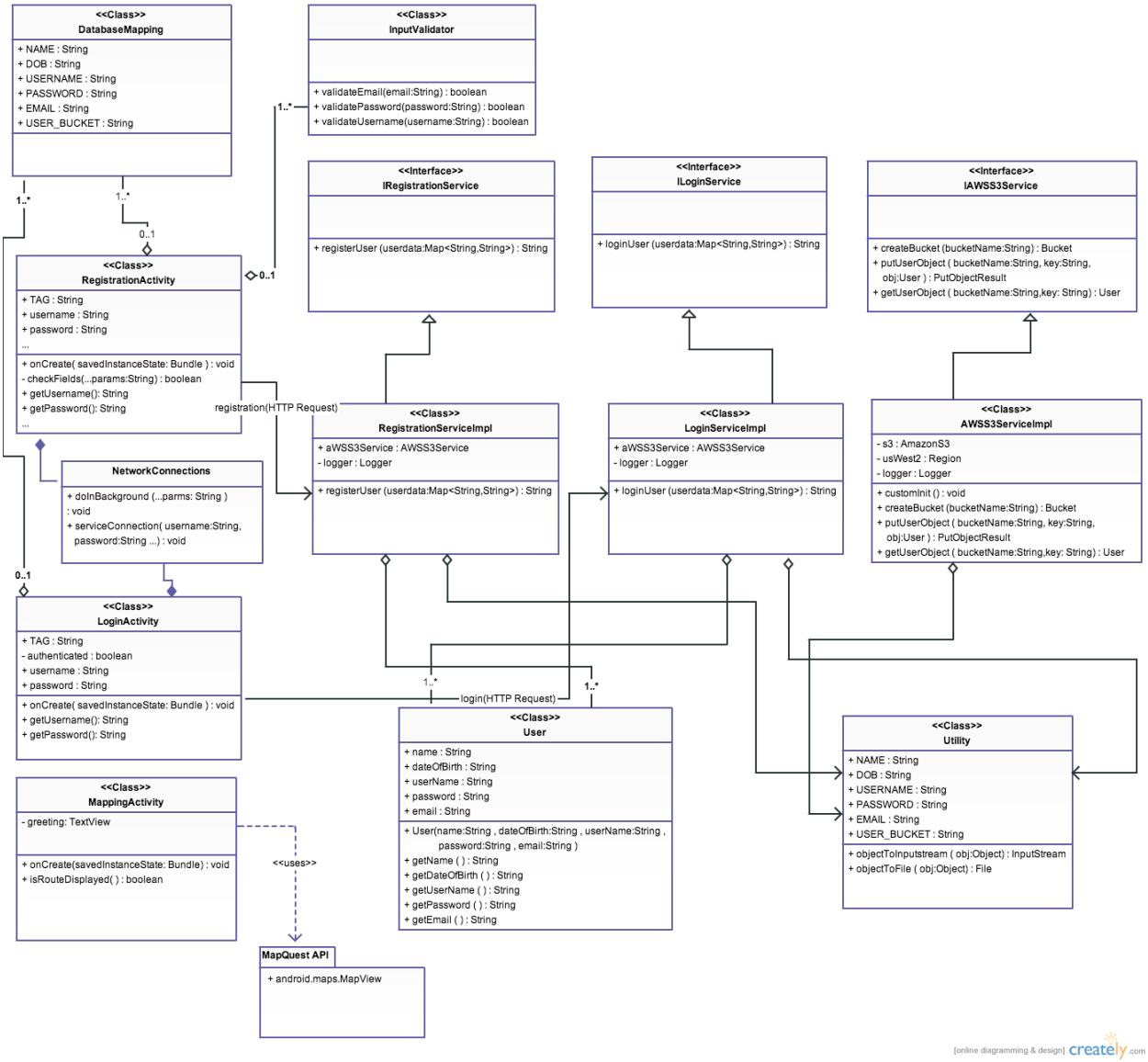

Figure 10 UML Diagram

4 Future Work and Features List

Using Agile Engineering the Photo Tracker Application will begin with basic functionality and evolve to support more features and capabilities. The first iteration of the project will be focused on getting the registration and login functionality up and running between the application and backend services. After a successful login, the user will see a greeting and a map. The next iteration would be focused on adding camera functionality and extracting geo-location from the images. The third iteration would focus on plotting the image locations on the map and adding the functionality to view images. Iterations would continue to add functionality until the prototype was completed for a first release. Future implementations would then be worked on for future versions of the application.

Some additional features to consider for future versions of the application include:

- The user shall be able to add friends
- The user shall be able to make a photo as public, private, or “friends only”
- The user shall be able to make their map public, private, or “friends only”
- The user shall be able to view friends photos [if marked public or “friends only”]
- The user shall be able to view friends map

These features were out of scope for version 1 and thus will not be implemented until a future version is developed.