Detailed Design Document

University of Southern California - CSCI 201

Team Members:

Izzy Benavente
Albert Yue
Sydney Liu
Rebecca Wu
Mingzhe (Elliscope) Fang

Mentor:

Jason Gui

PROJECT PROPOSAL

Instagram for food. Imagine being able to look at food. Make food. And eat food. In whatever situation you want. We envision a future where finding and making food is as easy as pulling out an app and searching. In the future we will build a robot that will automatically make the food using the instructions you choose. This is the first step in making it possible to have any food you want at any time without even touching a spoon!

The project will be done as a native android app with a full featured camera. It will essentially be a social media outlet where foodies and amateur cooks (just trying to get by) can post images of food and also add their own recipes which can be view by anyone looking into that particular dish.

GUI: We will be using the XML layout files to make a simple yet elegant interface where a user can easily interact with the app. We will be following Android Design guidelines, adhering to material design principles as closely as possible.

Multi-threading: We will incorporate commenting, rating, image-posting, and notification functionalities. Multi-threading will allow these to be performed quickly and at the same time.

Networking: Upon login, users will have their info checked against the user database on our server to authenticate their login. When users post images, post comments, and rate posts their data will be saved in our server and be put live on the app so other people can see their most recent additions and actions.

HIGH-LEVEL REQUIREMENTS:

We want a mobile Android application where users can upload and share images of their favorite dishes or recent meals to their accounts. Accounts may be created with a simple email and password login, and anyone with a profile should be able to like, share, comment, and rate other users' posts. If the user does not register or login, they will be able to view the app in a limited guest mode which allows them to browse posts without being able to vote, comment, or save them. This application should allow users to upload "noms", which are pictures and recipes of food, which may be taken directly from the app or uploaded from the user's phone gallery. However, users do not necessarily have to upload both images and recipes at the same time. To clarify, a nom is a post, which can have a recipe and images. In addition, each nom will be searchable by flavor, ingredients, recipe difficulty, and tags. Each nom will be expected to have relevant tags such as vegan, thai, noodles, etc. These tags will help make searches, categorization, and suggestions easier.

The app will have one-way following so anybody with an account can follow other users. Your "Food Feed" will show the latests posts from the people you follow. If you don't follow anyone, your Food Feed will just default to the most popular posts from the last 24 hours. Users will receive "NOMifications" whenever somebody begins following them or likes one of their noms. When viewing other noms, users will be able to vote on each other's noms, share them with their followers, bookmark them for future viewing, and also report any inappropriate posts. As one of the menu options, users can browse through noms and view them by different categories, even able to find dishes by looking for certain ingredients. There will also be a discovery menu option that lets users find new trending noms over the past day, week, and month.

TECHNICAL SPECIFICATIONS - Total 90 Hours

FUNCTIONALITY (82 hrs)

Registration and Login ***Total: 10 Hours***

UI for Login Page and Registration Page: ***4 Hours***

- The Login Page will have text fields and button to login
- The Registration Page will have text fields and a button to register

Functionality: ***6 Hours***

- Registration requires valid email, unique username, and password
- The password will be encrypted
- Login will send email and password to be compared against central server records
- Once authenticated, user will be sent to food feed page

Noms/Posts ***Total: 18 Hours***

UI for Noms/Posts: ***8 Hours***

- Posts will have a picture of food, ingredients used, and recipes.

Functionality: ***6 Hours***

- Noms will also suggest similar or related noms
- Users can upvote and comment on posts
    - Users can interact at the same time (multi-threading)

Commenting: **4 Hours**

- Only registered users can comment on noms.
- The most recent comments will be displayed at the top.

Search ***Total: 18 Hours***

By Keyword ***6 Hours***

- Users should also be able to search through Noms using a keyword search bar
- Results (Noms) should be returned on the same page

AutoComplete ***8 Hours***

· As the user types we'll check against the list of tags stored in our database and try to suggest words for the user to choose from. (Multi-threading)

By Category ***4 Hours***

· Users should be able to search through five categories: Vegetarian, Vegan, Breakfast, Lunch, and Dinner

- Results should be returned on the same page

Food Feed ***Total: 6 Hours***

- The app should default to this page on startup
- Users should be allowed to "Like" Noms
- Show the most popular posts by likes (multi-threaded updating)
    - Users must be logged in to like Noms

- Users who have not logged in should be directed to the login page when they click anything on this page (this includes Noms, “Like” buttons, the camera button, the profile button, and the notifications button)

Toolbar ***Total: 4 Hours***

- There will be a persistent toolbar at the bottom of the app where the user will be able to choose what page to view. The options are: Food Feed, Search, Camera, Notifications, and Profile.
- The toolbar which page you are currently on.
- Clicking on one of the options on the toolbar will take the user to the appropriate page.

Profile Page ***Total: 6 Hours***

- This page displays your profile picture and your username.
- The user will be able to modify account settings and log out from here.

Account Settings ***Total: 2 Hours***

- Users should be allowed to change their passwords

Photos ***Total:* 8 *Hours***

- Users should be able to upload photos from the phone gallery
    - Any photos taken from the gallery will be cropped to make the image square
- Users should also be able take photos using the app’s camera

Notifications ***Total: 8 Hours***

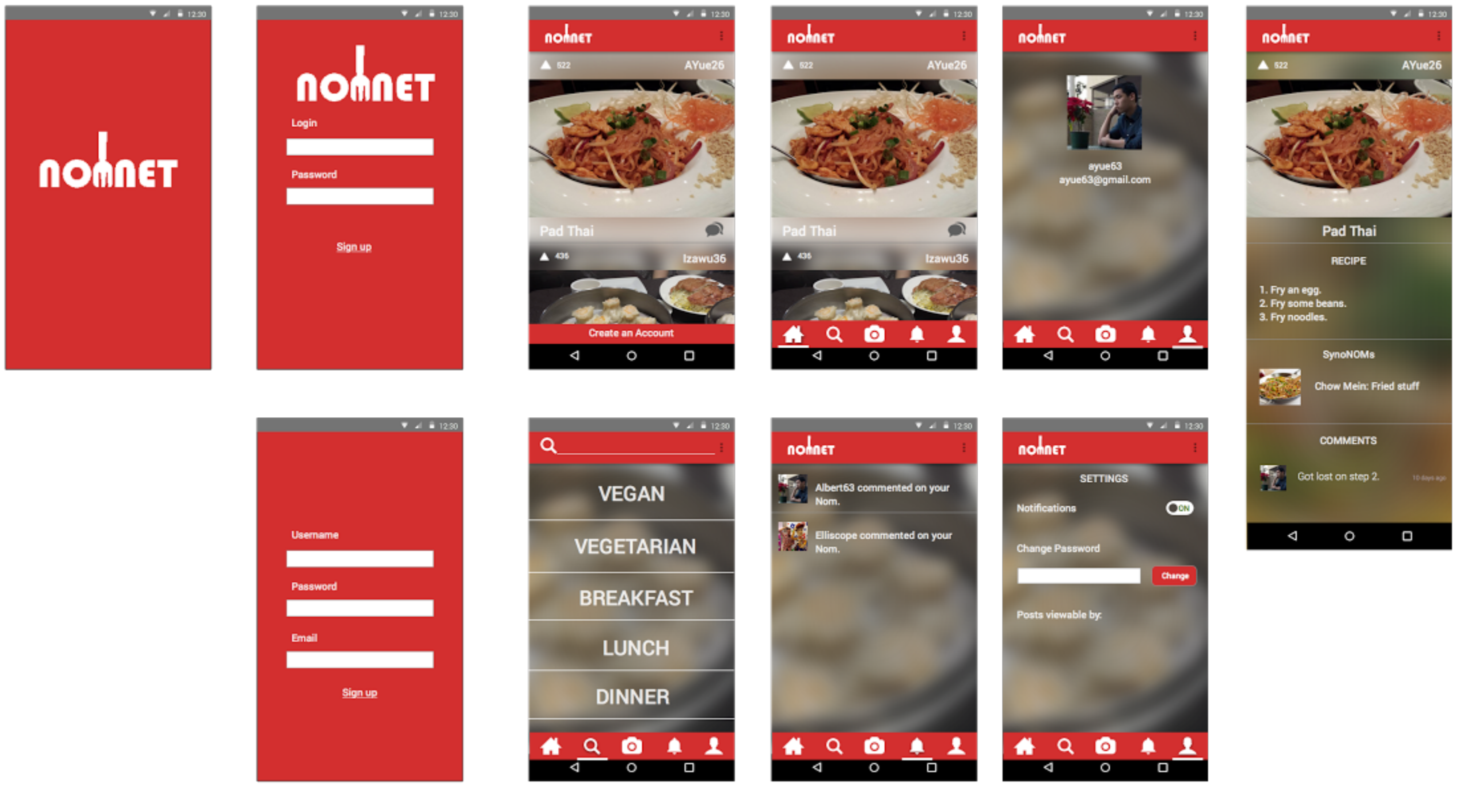
- Users should be notified when any of their posts receive comments
    - These will appear as push notifications as well as in the app

DATABASE (8 hrs)

Creation of Database Tables ***8 Hours***

- The database tables will have five tables: Users, Noms, Recipes, Images, and Comments
- The User table will be used for validating users and contains unique username, email, and password.
- The Nom table will be used for keeping track of user posts and contained post creator id, image id, unique nom id, number of likes. food name, recipe id, tags (array)
- The Recipe table will be used to store the different aspects of a recipe and will consist of dish name, ingredients (array), instructions, and image id.
- The Image table will be used to store all the images and will have the image file itself, unique id for image, and name.
- The Comments table will store all the comments and will have the comment text, the comment creator’s user id, a timestamp, and the id of the nom it was posted on.

UI/UX



1. Fry an egg.
2. Fry some beans.
3. Fry noodles.

Got lost on step 2.

Albert63 commented on your Nom.

Elliscope commented on your Nom.

DIFFICULT ALGORITHMS

Search Relevance

Users will be able to search through all the noms and users. Search will be done by querying the keywords in the search box. We will need an algorithm to determine the most relevant results that are returned from the query and then display them to the user.

Never-Ending Food Feed

The food feed will be populated with the top noms posted over the course of the past 24 hours. When the user's scroll hits the bottom, the app will load more noms to the feed. As the quantity of noms allows, this will be a never ending food feed that fills with noms as the user continues to scroll down.

CLASSES

Nom

- ATTRIBUTES:
  - ***User*** user
  - ***Recipe*** recipe
  - LinkedList<***Comment*** >allComents
  - String foodDesc
  - Image foodImg
  - String foodName
  - Int numUpvotes

- METHODS:
  - private User getCreator()
  - private Image getImage()
  - private void setImage()
  - private String getFoodName()
  - private void setFoodName(String fName)
  - private String getDesc()
  - private void setDesc(String desc)
  - private int getVotes()
  - private void createComment()
    - creates and adds new comment to the ***Nom***

User

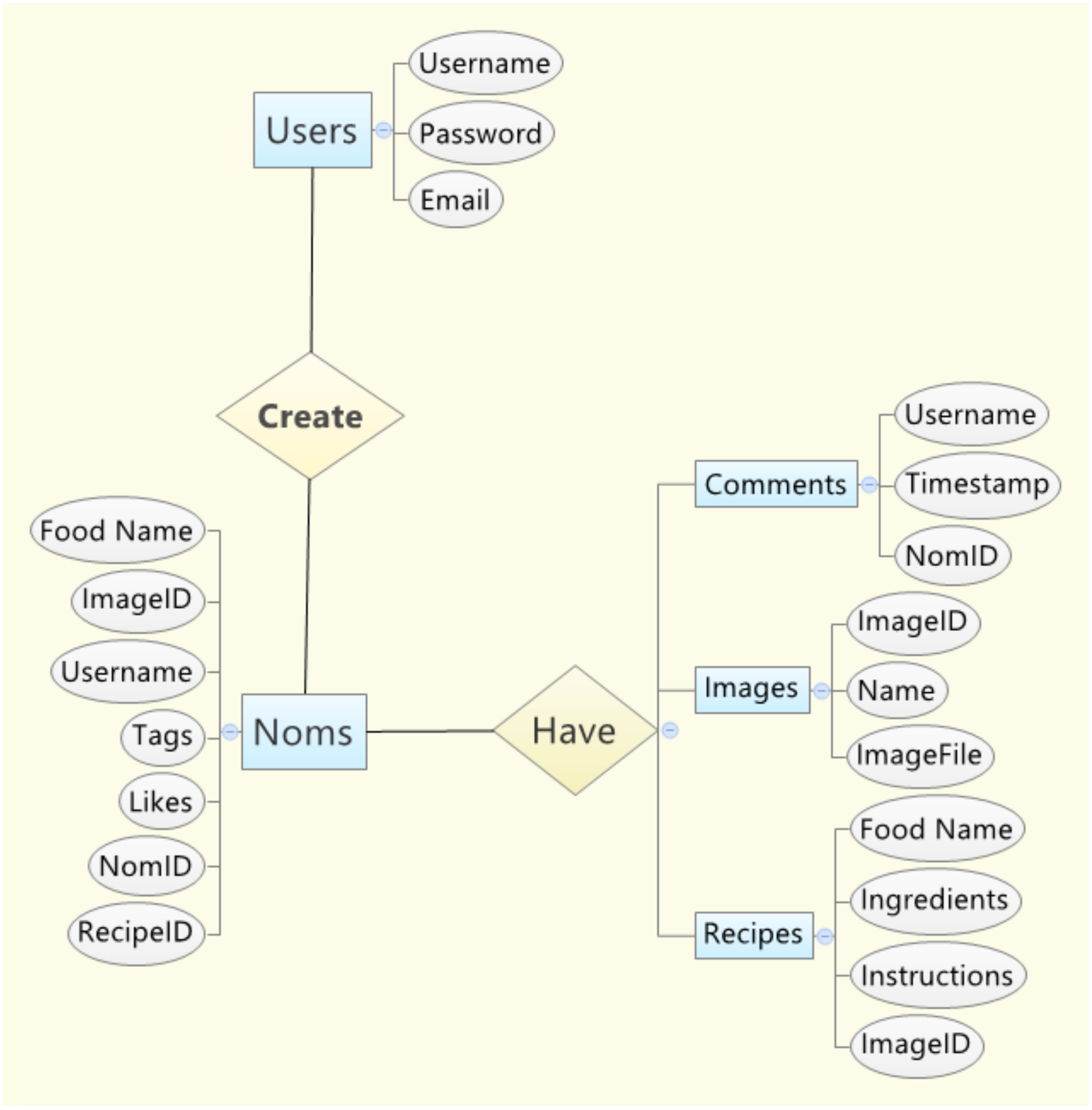
- ATTRIBUTES:
  - String userName
  - String realName
  - String password
  - String email
  - LinkedList<***Nom***> allNoms

- METHODS:
  - getUserName()
  - getRealName()
  - encrypt(String password)
  - decrypt(String password)

Recipe

- ATTRIBUTES
  - String ingredients
  - String instructions

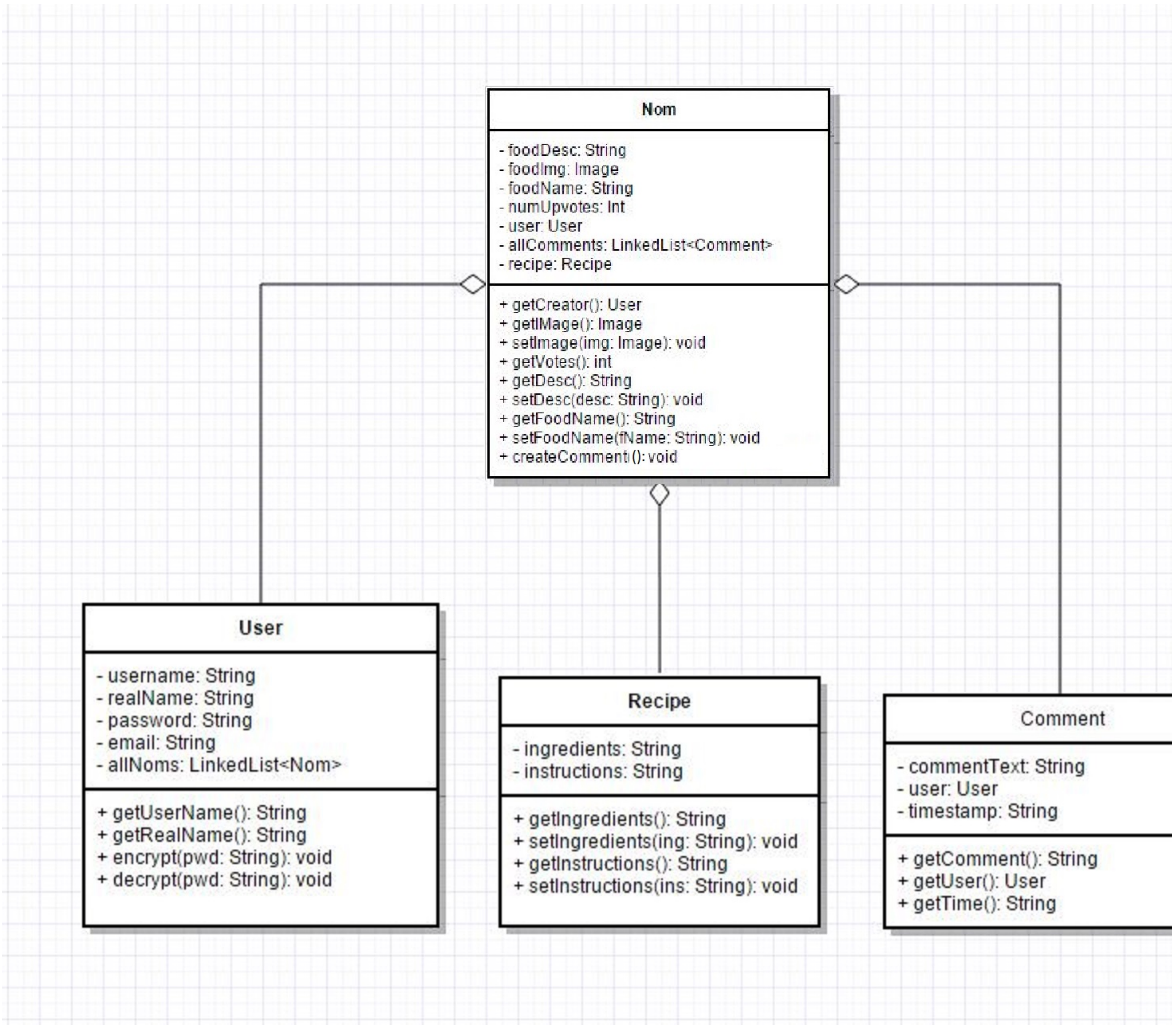
DATABASE SCHEMA (ER DIAGRAM)



HARDWARE/SOFTWARE REQUIREMENTS

The use of NomNet requires an Android mobile device that runs Android 4.1 Jelly Bean.

UML CLASS DIAGRAM



TESTING STRATEGIES

Stress Testing

One of the biggest complaints (and most frustrating experiences for users) is when an app crashes on a user. We will be stress testing to ensure that regardless of what device, you will have a great experience on the app.

For stress testing, we will use Exerciser Monkey (http://developer.android.com/tools/help/monkey.html), which generates random touches, clicks, and more. This will help us recognize when the app freezes, receives any un-handled exceptions, and if the app crashes with the intense and random usage from Exerciser Monkey. It will also try many other cases such as turning off wifi, cell service, muting, and more to ensure that that all inputs are tested. The program will run thousands of outputs quickly. There is a really nice tutorial about how to use it here: http://adventuresinqa.com/2014/04/01/how-to-stress-test-your-android-app-with-monkey/

Monkey is able to put the computational load necessary to test the app.

We may consider options of load testing as well as a side-test of the stress testing. While we do not anticipate loads to be a huge issue, we may take a look at this. Load testing seems to be primarily done with other services that connect through your app and emulate heavy loads and connections while tracking usability.

Regression Testing

As we code and make changes, we will keep track of any major bugs and create fixes for them. When large changes are made to the program, we will test to ensure that the documented major bugs do not reappear in our newer versions. If for any reason we need to switch to a newer version of Android, we will perform configuration testing to see if our program still works on the targeted devices that run our targeted version of Android and newer. We will perform this test by running functionality tests on different, emulated versions of Android devices and record the results, fixing any unexpected bugs.

To ensure optimal performance of our app, we will monitor the execution time of certain functions such as Food Feed population and searching and test them at each major update to the app to maintain a certain level of quality and speed. As mentioned in the Stress Test section, we will use Exerciser Monkey to see how the app handles random input. We will also perform this test as we update the app to check for any new bugs.

Black Box Testing

We will assign testers to parts of the program which other members have coded in order to avoid any testing bias. Because the programmers of a certain section will expect it to perform the functionality they coded it for, we need to test code with which we are unfamiliar. Our team will perform black box testing and record any complications or bugs discovered along the way. We will then pass our findings to the person that coded that portion so the bug can be fixed.

We will be using black box testing to ensure we are meeting the project requirements our team has put together. This will help us validate that we are creating the program with the functionality we planned.

White Box Testing

The benefit of white box testing is that the tester will have knowledge of the code as they run the tests. We will have the coders of each section run white-box tests on their own code. Because they know what to expect from the results, they will know what tests to run against their code's functionalities.

White box testing will help us verify that the program performs its functionalities correctly and produces the proper, expected output.

Planned Unit Tests

At the moment, we do not have a functional application that we can test. As a result, we will be creating test cases and expected results, but we will add the actual results once we have a working application. We will follow the the four-item test case template (ID, Description, Expected Results, Actual Results) when planning your test cases. However,we will add Coder, Tester, and Actual Results columns when we implement the actual tests. This will allow us to effectively test the functionality of our code and ensure that we achieve the results that we want.

Test ID	Description	Expected Results

1	User inputs a valid email address and password when creating account.	User account creation is successful and he or she moves on to the login screen.
2	Upon startup, (guest) user is taken to the feed showing most popular Noms. User then clicks on a registered user button, such as Nom, upvote, or toolbar item.	User is taken to Sign Up page.
3	User makes a login attempt with an invalid email address or password.	App should notify user that the email address was not valid.
4	User uploads non-image file to Nom.	Error toast message pops up explaining the file was unable to be uploaded.
5	User makes a successful login attempt.	User should be taken to the Food Feed page.
6	User searches for a food dish.	The app should display relevant noms that match the keywords used in the search. If no such noms exist, prompt the user to try different keywords.
7	Successfully logged-in user upvotes a Nom.	A notification is sent to the Nom's owner (in the app and on the phone's notification bar if allowed), the upvote button on the user's screen turns red, and the upvote count increases by one.
8	User comments on a nom.	The page should refresh to show the new comment as the most recently made comment on the nom. It should should the name of the commenter and the time posted.
9	User chooses one of the pre-made	The app should take the user to a page that

	categories: Vegan, Vegetarian, Breakfast, Lunch, Dinner	has the most popular dishes of that category.
10	User tries to change password.	Text boxes prompt for old password, new password, and new password again. If the new passwords don’t match, the text boxes are cleared, and a toast message appears saying the new passwords do not match. If the new password is the same as the old password, the new password is cleared from the two boxes, and a toast message appears saying the user must enter a different password.
11	User B comments on a User A’s nom.	User A should see a notification indicator on the bottom toolbar.
12	User has a notification indicator and presses the notification button on the toolbar.	The notification indicator on the toolbar should disappear and the notifications page should show the user the most recent notifications they’ve received.
13	User B comments on User A’s nom while User A’s NomNet app is not open..	User A should receive a push notification that tells them about User B’s comment.
14	User drags down on a food feed	Feed is refreshed with updated information.
15	User switches posts to private.	None of this user’s posts are visible to other users.
16	User taps camera icon	Opens a menu with 2 tabs: Gallery and camera. Users can choose to upload pictures from the gallery or take a new picture here.
17	User inputs an invalid email	User is asked to please input correct

	address and password when creating account.	information.
18	User types in part of a Nom	The search bar should attempt to complete the Nom and display suggestions based on the partial string.
19	User is disconnected from wifi and tries to log in.	Toast will appear saying the app cannot authenticate the user.
20	User is disconnected from wifi and tries to interact with the app functionalities.	Already loaded Noms from when user was connected to wifi will appear, but no new ones will be able to load and toast message will appear saying the feed cannot be loaded.

DEPLOYMENT INSTRUCTIONS

We are using Android Studio to develop our app, and as a result we will need to make a few adjustments before releasing it to the public. We will take the following steps below in order to ship it to the Google Play store:

-NomNet requires an Android mobile device that runs Android 4.1 Jelly Bean-

Code Cleaning

1. Remove all Log calls in our code
2. In the AndroidManifest.xml set android:debuggable="false"
3. Insert values for android:versionCode and android:versionName (located in the <manifest> element)

Project Cleaning

1. Double check our jni/, lib/, and src/ directories and make sure they contain the appropriate file types.
2. Remove unused datafiles, such as files in the res/ directory
3. Check lib/ directory for test libraries and remove unnecessary or unused libraries
4. Review our assets/ directory, checking the res/raw directory for raw assets, and removing unnecessary or unused files.

Gradle Build Settings

1. Set the necessary <uses-permission> element
2. Specify values for android:icon and android:label (in <application> element)

External Servers

1. You can see what we did with node.js by going to: http://ec2-52-8-95-32.us-west-1.compute.amazonaws.com:3000/b
2. http://ec2-52-8-95-32.us-west-1.compute.amazonaws.com:3000/a pulls a mysql query (from the server, but can't be seen)
3. http://ec2-52-8-95-32.us-west-1.compute.amazonaws.com:3000/ uses some sockets
4. To ssh into our server to see what we did:
    a. Go to and get admin.pem
    b. IN THE SAME DIRECTORY AS THIS FILE:
        i. sudo chmod 600 admin.pem
        ii. ssh -i admin.pem ubuntu@ec2-52-8-95-32.us-west-1.compute.amazonaws.com
    c. To see our MYSQL:
        i. Type: mysql -u root -p
        ii. No password
        iii. Type: use Nomnet

        iv. You can see our database by using some basic SQL commands.

    d. To see our node code. Go to the nomnet directory. app.js contains most of the relevant code

<u>Set Application Version</u>

1. In the manifest, add the element <uses-sdk>
    a. Set the androidminSdkVersion attribute to Android 4.1 Jelly Bean

<u>Signing and Exporting Project from Android Studio</u>

1. In the top menu, go to Build>Generate Signed APK
2. Click Create New and fill out the signing details, then press OK
3. Complete information for keystore, private key, and passwords, then hit Next
4. Choose a destination for the completed APK and press Finish

<u>Installing APK on Android</u>

1. Transfer file to Android File Systems through USB, Google Play, or email.
2. Go to File Systems or Downloads, depending on how you imported the APK. If email, go to the downloads folder. If from USB, see where the file was imported in the file systems. If Google Play, it automatically installs. If your operating system does not have a default file system, you can find one on the Google Play Store. Launch APK on phone to install/launch by touching on the APK. It will pull up a installer and you will be able to install the APK.

RECENT CHANGES

Changes to High Level Requirements Document

- We have decided to remove the ability to follow people because we realized that this would create a more complicated interconnected network that we may not complete in time. As a replacement, we repurposed the “Food Feed” to be a never ending list of noms which are the top voted noms of the past 24 hours.

    - We put a great deal of thought into this change and ultimately decided we needed to lessen the workload due to an unforeseen circumstance where our team member Elliscope has hospitalized for most of the semester due to complications in his lungs.

Changes to Technical Specification Document

- We made our technical specification document with our modified Food Feed in mind which results in an approximation that the project will take 90 hours to complete instead of our initial estimate which was over 100 hours.

Addition of Testing Strategies Document 4/2

Addition of Deployment Instructions 4/12

Addition of Deployment Instructions 4/26:

Node js backend can be seen now. Check how to use above