PubWC Bathroom Review App
By Clay Jacobs
Advisor: Alexander Dekhtyar
Computer Science Department
California Polytechnic State University
2017

Abstract

For my senior project, I developed an iOS application to allow users to find, rate, and review nearby public restrooms. The app takes advantage of crowdsourced data to collect bathroom and review information. I also created a REST API to interface with the backend database that could be used to port the application to other platforms.
# Table of contents

1 Introduction 1
   1.1 The Problem 1
   1.2 Background 1
      1.2.1 SitOrSquat 1
      1.2.2 Toilet Finder 1
      1.2.3 Flush - Toilet Finder & Map 1
   1.3 Development Timeline 2

2 Design 3
   2.1 Overview 3
   2.2 App 3
   2.3 Backend 4
      2.3.1 Accounts Table 5
      2.3.2 Buildings Table 6
      2.3.3 Bathrooms Table 7
      2.3.4 Amenities Table 8
      2.3.5 Reviews Table 9
      2.3.6 Notes Table 10
      2.3.7 Functions and Triggers 11
   2.4 Challenges 12

3 Application Overview 13
   3.1 Overall Scenario 13
   3.2 User Account Scenarios 14
      3.2.1 Registering a new account 14
      3.2.2 Logging into and out of an account 15
      3.2.3 Adding a profile picture 16
      3.2.4 Updating an account 17
      3.2.5 Deleting an account 18
   3.3 Bathroom and Review Creation Scenarios 19
      3.3.1 Viewing existing buildings 19
      3.3.2 Adding a building 20
      3.3.3 Viewing existing bathrooms 21
      3.3.4 Adding a bathroom 22
Table of figures

| Figure 2.1 - Application Design Diagram | 3 |
| Figure 2.2 - Servlet URLs              | 4 |
| Figure 2.3 - Accounts Table Create Statement | 5 |
| Figure 2.4 - Buildings Table Create Statement | 6 |
| Figure 2.5 - Bathrooms Table Create Statement | 7 |
| Figure 2.6 - Amenities Table Create Statement | 8 |
| Figure 2.7 - Reviews Table Create Statement | 9 |
| Figure 2.8 - Notes Table Create Statement | 10 |
| Figure 2.9 - Vincenty Formula Function | 11 |
| Figure 2.10 - Change User Trigger       | 11 |
| Figure 2.11 - Update Username Trigger   | 11 |
| Figure 3.1 - Initial Login View         | 14 |
| Figure 3.2 - Initial Registration View  | 14 |
| Figure 3.3 - Registration With Error    | 14 |
| Figure 3.4 - Registration Success       | 14 |
| Figure 3.5 - Filled In Login View       | 15 |
| Figure 3.6 - Login View With Error      | 15 |
| Figure 3.7 - Initial Profile View       | 15 |
| Figure 3.8 - Profile Picture Selection From Photo Library | 16 |
| Figure 3.9 - Profile View With Profile Picture Selected | 16 |
| Figure 3.10 - Initial Update Profile View | 17 |
| Figure 3.11 - Filled In Update Profile View | 17 |
| Figure 3.12 - Update Profile View With Error | 17 |
| Figure 3.13 - Updated Profile View      | 17 |
| Figure 3.14 - Attempting Login After Deleting Account | 18 |
| Figure 3.15 - Building Map View         | 19 |
| Figure 3.16 - Building List View        | 19 |
| Figure 3.17 - Initial Add Building View | 20 |
| Figure 3.18 - Select Custom Location View | 20 |
| Figure 3.19 - Filled In Add Building View | 20 |
| Figure 3.20 - Add Building View With Error | 20 |
| Figure 3.21 - Building Detail View Overlay | 21 |
| Figure 3.22 - Current Building View     | 21 |
| Figure 3.23 - Initial Add Bathroom View | 22 |
| Figure 3.24 - Filled In Add Bathroom View | 22 |
| Figure 3.25 - Add Bathroom View With Error | 22 |
Introduction

1.1 The Problem

While location based review services aren’t a new idea, there isn’t a really great review application for services like public restrooms. To solve this problem, I developed a crowdsourced iOS application to allow users to find, rate, and review nearby public restrooms. I also created a REST API to interface with the backend database that could be used to port the application to other platforms.

1.2 Background

While some similar apps already exist on the Apple App store and Google play store, these apps mostly support a very limited set of features. Very few are cross platform apps and many lack the ability to edit locations. Adding, updating, and deleting bathroom data in a review app is a very important feature since location data can become stale over time and needs to be easy to modify. While many apps show nearby bathroom locations, many also have a very limited amount of data on each bathroom. A user may need a bathroom for different reasons at any given time, so distance is just one of many criteria that a user might care about in finding a bathroom. Existing bathroom review apps have very limited data models for bathrooms, which also limits the criteria users can search for within the app. I tried several of the existing apps, some of which are described further below.

1.2.1 SitOrSquat

The SitOrSquat app is arguably the best bathroom review app currently available on the App Store. While the old SitOrSquat app had it’s problems, the new version dramatically improved the user experience. The SitOrSquat app allows logged in users to add and delete bathrooms. In addition SitOrSquat provides a much better interface for search, search filtering, and browsing bathroom review data than competitors, even providing bathroom photos. However, the SitOrSquat app doesn’t have as much bathroom data associated with it than other apps. The database for SitOrSquat contains over 100,000+ restrooms.

1.2.2 Toilet Finder

Toilet Finder is designed mainly as a bathroom locator app. Toilet finder does have slightly more bathroom data and does include the ability to flag bathrooms that no longer exist. This app also gives directions to the bathroom, which is a more common feature amongst the locator apps. Toilet Finder also has photo’s and the app has over 130,000 restrooms in it’s database. Toilet finder lacks search functionality and doesn’t have quite as much bathroom data as SitOrSquat in less populated areas like San Luis Obispo.

1.2.3 Flush - Toilet Finder & Map

Flush - Toilet Finder & Map primarily functions as a locator app. The app provides bathroom addresses and the ability to rate the bathroom. The app does provide directions to the bathrooms via the native Apple Maps app. The app also shows whether a bathroom has a fee,
key access, and disabled access. Flush has the least data associated per bathroom of these three apps. The database for Flush - Toilet Finder & Map contains over 190,000 restrooms.

1.3 Development Timeline

During Spring quarter I worked on setting up the server, REST API, and the accounts section of the app. I also started working on the WC editor section. Over the summer I finished the WC editor section and started working on polishing the UI. During fall quarter I built the WC locator section and finished the UI work.
2 Design

2.1 Overview

The project consists of an app that calls a REST API backend. Each component is shown in the diagram below and is discussed in more detail in the following sections.

2.2 App

The mobile application is built for iOS and uses Swift. The app is designed according to the MVC protocol, where each view has its own view controller class. I also used Xcode’s interface builder to design the UI.
2.3 Backend

The backend consists of a REST API and a MySQL database. The database is hosted on an Amazon EC2 instance running a Tomcat web server. The EC2 instance is behind an Elastic Load Balancer. The Load Balancer provides a front facing DNS name for domain name association, a layer of access control for ports, and a means of associating a SSL certificate, allowing URLs to be https.

The REST API endpoint URLs are written as Java HttpServlets and return results in JSON format. Parameters are passed in via URL in get requests. The REST API endpoint URLs are listed below.

![Server URLs](image)

The database consists of 6 tables and several useful functions and triggers, each of which will be described below.
2.3.1 Accounts Table

Accounts represent the app users. Creating an account allows users to generate content, not just consume it. The create table statement is shown below.

```sql
CREATE TABLE `PubWCAccounts` (  `userId` int(11) NOT NULL AUTO_INCREMENT,  `userType` tinyint(1) DEFAULT '0',  `email` varchar(120) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  `password` varchar(180) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  `firstName` varchar(40) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  `lastName` varchar(80) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  `username` varchar(80) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  PRIMARY KEY (`userId`) );
```

- userId - id for the user
- userType - type of user account (e.g. admin, basic user, etc.)
- email - email address associated with the account
- password - account password encrypted
- firstName - user’s first name
- lastName - user’s last name
- username - username for the account
2.3.2 Buildings Table

Buildings represent actual buildings in the real world. A building can contain multiple bathrooms. The create table statement is shown below.

```
CREATE TABLE `PubWCBuildingLocs` (  
  `buildingID` int(11) NOT NULL AUTO_INCREMENT,  
  `creatorID` int(11) NOT NULL DEFAULT '4',  
  `latitude` float DEFAULT NULL,  
  `longitude` float DEFAULT NULL,  
  `name` varchar(120) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `type` varchar(80) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `floors` int(3) NOT NULL,  
  PRIMARY KEY (`buildingID`)  
);
```

FIGURE 2.4 - BUILDINGS TABLE CREATE STATEMENT

- buildingID - unique id for the building
- creatorID - unique id for the user account that created the building
- latitude - latitude that the building is located at
- longitude - longitude that the building is located at
- name - name of the building
- type - type of the building (e.g. food, school, recreation, etc.)
- floors - number of floors in the building
### 2.3.3 Bathrooms Table

Bathrooms represent the bathrooms themselves. The create table statement is shown below.

```
CREATE TABLE `PubWC.Bathrooms` (  
  `bathroomID` int(11) NOT NULL AUTO_INCREMENT,  
  `buildingID` int(11) NOT NULL,  
  `name` varchar(80) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `floor` int(3) NOT NULL,  
  PRIMARY KEY (`bathroomID`),  
  KEY `fk_bathrooms_buildings` (`buildingID`),  
  CONSTRAINT `fk_bathrooms_buildings` FOREIGN KEY (`buildingID`) REFERENCES `PubWC.BuildingLocs` (`buildingID`) ON DELETE CASCADE
);```

- **bathroomID** - unique id for the bathroom
- **buildingID** - unique id for the building
- **name** - name for the bathroom
- **floor** - floor in the building that the bathroom is located on.
2.3.4 Amenities Table

Amenities are abstract items and features associated with a bathroom, including gender, toilets provided, supplies, and facilities. The create table statement is shown below.

```
CREATE TABLE 'PubWC_Amenities' (
    'amenityID' int(11) NOT NULL AUTO_INCREMENT,
    'bathroomID' int(11) NOT NULL,
    'featureType' int(11) NOT NULL,
    'name' varchar(80) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    'amenityCount' int(4) NOT NULL,
    PRIMARY KEY ('amenityID'),
    KEY 'fk_amenities_bathrooms' ('bathroomID'),
    CONSTRAINT 'fk_amenities_bathrooms' FOREIGN KEY ('bathroomID') REFERENCES 'PubWC_Bathrooms' ('bathroomID') ON DELETE CASCADE
);
```

- amenityID - unique id for the amenity
- bathroomID - unique id for the bathroom
- featureType - type of amenity
- name - name of the amenity
- amenityCount - number of the amenity in question
2.3.5 Reviews Table

Reviews are user provided ratings and text reviews of specific bathrooms. The create table statement is shown below.

- reviewID - unique id for the review
- bathroomID - unique id for the bathroom being review
- reviewerID - unique id for the author of the review
- rating - rating given for the bathroom
- review - review given for the bathroom
- reviewer - username of the author of the review
- title - title of the review for the bathroom
2.3.6 Notes Table

Notes provide a means to make posts that may be separate from bathroom related content. Info notes are the most general category of notes and event notes can be used for providing bathroom info for specific events. The create table statement is shown below.

```
CREATE TABLE `PubWCNotes` (  
`noteID` int(11) NOT NULL AUTO_INCREMENT,  
`authorID` int(11) NOT NULL,  
`authorUsername` varchar(80) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
`latitude` float DEFAULT NULL,  
`longitude` float DEFAULT NULL,  
`noteTitle` varchar(120) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
`note` varchar(500) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
`noteType` int(2) DEFAULT NULL,  
`datePosted` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',  
PRIMARY KEY (`noteID`)
);
```

FIGURE 2.8 - NOTES TABLE CREATE STATEMENT

- `authorID` - unique id for the author of the note
- `authorUsername` - username of the author of the note
- `latitude` - latitude that the note is located at
- `longitude` - longitude that the note is located at
- `noteTitle` - title of the note
- `note` - the note itself
- `noteType` - the type of note (e.g. info, event)
- `datePosted` - the date the note was posted on
2.3.7 Functions and Triggers

The database also contains several functions and triggers. To calculate distances from GPS latitude and longitude coordinates, I implemented the vincenty formula as shown below. I also created triggers on delete and update for username across the app. When users accounts are deleted, review and bathroom information they submitted will be kept under a different user account to preserve app data that could be useful to other users.

```
DELIMITER $$
DROP FUNCTION IF EXISTS vincenty$$
CREATE FUNCTION vincenty(
  lat1 FLOAT, lon1 FLOAT,
  lat2 FLOAT, lon2 FLOAT
) RETURNS FLOAT
NO SQL
DETERMINISTIC
COMMENT 'Returns the distance in degrees on the Earth between two known points of latitude and longitude using the Vincenty formula from http://en.wikipedia.org/wiki/Great-circle_distance'
BEGIN
  RETURN DEGREES(
    ATAN2(
      SQRT(
        POW(COS(RADIANS(lat2))*SIN(RADIANS(lon2-lon1)),2) +
        POW(COS(RADIANS(lat1))*SIN(RADIANS(lat2)) -
        (SIN(RADIANS(lat1))*COS(RADIANS(lat2)) *
        COS(RADIANS(lon2-lon1))),2),
      SIN(RADIANS(lat1))*SIN(RADIANS(lat2)) +
      COS(RADIANS(lat1))*COS(RADIANS(lon2-lon1)))
    )
  )
END$$
DELIMITER ;
```

FIGURE 2.9 - VINCENTY FORMULA FUNCTION

```
Delimiter $$
create trigger changeUserTrigger
before delete on PubWCAccounts
for each row begin
  update PubWCReviews, (select username from PubWCAccounts where userID = 4) A set reviewerID = 4, reviewer = A.username where old.userID = PubWCReviews.reviewerID;
  update PubWCBuildingLocs set creatorID = 4 where old.userID = PubWCBuildingLocs.creatorID;
end $$
Delimiter ;
```

FIGURE 2.10 - CHANGE USER TRIGGER

```
Delimiter $$
create trigger updateUsernameTrigger
after update on PubWCAccounts
for each row begin
  update PubWCReviews, (select username from PubWCAccounts where userID = new.userID) A set reviewer = A.username where new.userID = PubWCReviews.reviewerID;
  update PubWCNotes, (select username from PubWCAccounts where userID = new.userID) A set authorUsername = A.username where new.userID = PubWCNotes.authorID;
end $$
Delimiter ;
```

FIGURE 2.11 - UPDATE USERNAME TRIGGER
2.4 Challenges

Before doing this project, I had very little experience using Autolayout constraints to adapt my application to multiple screen sizes. By the end of this project, I became very good at using Autolayout constraints to build my application’s UI. I also hadn’t worked much with handling location data or communication between a server and a mobile application. This project gave me experience doing both tasks. While these challenges were all problems I had to overcome during development of the app, they also made the project more interesting and enjoyable to complete.
3 Application Overview

3.1 Overall Scenario

Suppose a new user wants to review a new bathroom in the Cal Poly SLO library upon finding this bathroom doesn't exist in the app. They create a new user account, then add the new bathroom and write a review for it. Later they want to update part of the bathroom's data and make a correction, find the bathroom again, and create a note.
3.2 User Account Scenarios

3.2.1 Registering a new account

When the user first opens the app, they will be presented with the login view shown in figure 3.1. When the Register button is tapped, the user will be taken to the registration view shown in figure 3.2. After filling in the text fields in figure 3.2, tapping Register will create a new account. If any registration data is missing or incorrect, an error message will be displayed as shown in figure 3.3 describing the registration error. Otherwise, a message saying, “The user has been registered!” will be presented to the user as shown in figure 3.4.
3.2.2 Logging into and out of an account

The user can access the account login view (figure 3.1) from the registration view (figure 3.2) by tapping the Login button, or as the initial view in the accounts section of the app if the user has not already logged in. The user can tap the Login button to log into their account after filling out the text fields in the login view as shown in figure 3.5. If any login information in figure 3.5 is missing or incorrect, an error message describing the problem will be presented as shown in figure 3.6. Otherwise the user will be presented with the profile view shown in figure 3.7. To log out of their account, the user can tap the Logout button in the top left of the profile view in figure 3.7 and be taken back to the initial login view shown in figure 3.1.
3.2.3 Adding a profile picture

A profile picture can be added by tapping the *Update Profile Picture* button in the profile view as seen in figure 3.7. The user may be asked for permission to access the photo library if the app has not done so already. Once granted, the user can select their desired profile picture from their photos or press cancel as shown in figure 3.8. Once a picture is selected, it will be shown in place of the default placeholder profile picture as shown in figure 3.9.
3.2.4 Updating an account

Users can update their profile data by pressing the *Edit* button at the top right of the profile view shown in figure 3.7. The user will then be taken to the update account view shown in figure 3.10, which will already be populated with the current user account’s data. After making the updates as shown in figure 3.11, the user can press the *Save* button to submit the changes or the *Back* button to cancel the update. If nothing is updated or an error in the updated data is detected, an error will be displayed as shown in figure 3.12. Otherwise the user will be returned to the updated profile view as shown in figure 3.13.
3.2.5 Deleting an account

The user can delete an existing account from the profile view by tapping the *Delete Account* button at the bottom of figure 3.7. This will return the user to the initial login view in figure 3.1. Attempting to log into the deleted account will now result in an error as shown in figure 3.14.
3.3 Bathroom and Review Creation Scenarios

3.3.1 Viewing existing buildings

Users can view buildings for editing by navigating to the WC Editor tab in the tab bar controller. Buildings can be viewed both a map view and a list view format as shown in figure 3.15 and figure 3.16 respectively. For the building map view, the user may be asked for permission to access the users current location if the app has not done so already. Users can navigate between the two views by tapping the View as List button in figure 3.15 and the View as Map button in figure 3.16. The grey and blue dot in the map view shows the users current location while the green map icons show the locations of bathrooms on the map. The Center button in the top left of the buildings map view in figure 3.15 will center the map view on the users current location if this is no longer the case. The Reload button in the top right of the buildings map view in figure 3.15 will reload the bathroom data from the server and refresh the map view if needed. Both the building map view and building list view will only load the 10 nearest bathrooms first. To view more bathrooms, the Next 10 and Previous 10 buttons can be used to navigate search results in the map view seen in figure 3.15. A Load more cells button can be tapped to load an additional 10 buildings in the list view seen in figure 3.16.
3.3.2 Adding a building

A new building can be added as a logged in user by tapping the Add Building button from the building map view in figure 3.15 or the building list view in figure 3.16. The user will be taken to the add building view shown in figure 3.17. The default latitude and longitude and the ones used when the Refresh Location button in figure 3.17 is tapped correspond to the user's current location. When the Select Custom Location button is tapped the user is taken to the custom location selection view shown in figure 3.18. The latitude and longitude coordinated values at the bottom of figure 3.18 in the map view represent the custom location that will be used for the new building once the Use button is tapped. The custom location is the location at the center of the current map view region. When all the building data is filled in as shown in figure 3.19, the user can tap the Add button to create a new building. If any data is missing or incorrect in figure 3.19 an error will be displayed describing the error as shown in figure 3.20. Otherwise the user will be taken back to the updated originating view, either the building map view in figure 3.15 or the building list view in figure 3.16.
3.3.3 Viewing existing bathrooms

Bathrooms can be viewed within the WC Editor tab as part of the building view. From the building map view in figure 3.15, tapping the map icon for the building in question will show a new overlay view for the building as shown in figure 3.21. When the View button in figure 3.21 is tapped, the user will be taken to the corresponding building view shown in figure 3.22. From the building list view in figure 3.16, tapping the View button for the building in question will navigate to the corresponding building view shown in figure 3.22. The building view shows the name of the current building as well as a list of the bathrooms in that building organized by floor.
3.3.4 Adding a bathroom

A new bathroom can be added as a logged in user by tapping the Add Bathroom button at the bottom of the current building view as shown in figure 3.22. The user will be taken to the add bathroom view shown in figure 3.23. After filling out the fields and selecting from the dropdown menus in the add bathroom view, the user can tap the Add button to create the new bathroom as shown in figure 3.24. If any data is missing or incorrect in figure 3.24 an error will be displayed describing the error as shown in figure 3.25. Otherwise the user will be taken back to the updated originating building view from figure 3.22.
3.3.5 Viewing existing amenities and reviews

Amenities and reviews can be viewed within the WC Editor tab in the bathroom amenities view and bathroom reviews view respectively. From the building view in figure 3.22, tapping the View button for the bathroom in question will navigate to the bathroom amenities view as shown in figure 3.26 or the bathroom reviews view as shown in figure 3.27 for the bathroom in question. Both views will show the name of the current bathroom at the top of the view. To navigate between both views, the user can tap the Reviews button in figure 3.26 and the Amenities button in figure 3.27. To view all of a longer review, tap the down arrow button on the right under the review as shown in figure 3.27.
3.3.6 Adding an amenity

A new amenity can be added as a logged in user by tapping the *Add Amenity* button at the bottom of the bathroom amenities view as shown in figure 3.26. The user will be taken to the add amenity view shown in figure 3.28. The next fields on this page will appear after the previous field has been set to a value on this page. After filling out the fields and selecting from the dropdown menus in the add amenity view, the user can tap the *Add* button to create the new amenity as shown in figure 3.29. If any data is missing or incorrect in figure 3.29 an error will be displayed describing the error as shown in figure 3.30. Otherwise the user will be taken back to the updated originating bathroom amenities view from figure 3.26.
3.3.7 Adding a review

A new review can be added as a logged in user by tapping the Add Review button at the bottom of the originating bathroom reviews view as shown in figure 3.27. The user will be taken to the add review view shown in figure 3.31. After filling out the fields add review view, the user can tap the Add button to create the new review as shown in figure 3.32. If any data is missing or incorrect in figure 3.32 an error will be displayed describing the error as shown in figure 3.33. Otherwise the user will be taken back to the updated originating bathroom reviews view from figure 3.27.
3.4 Bathroom and Review Editing Scenarios

3.4.1 Updating a building

Buildings can be updated as a logged in user by tapping the *Update* button at the top right of the current building view as shown in figure 3.22. The user will be taken to the update building view filled out with the current building data filled in as shown in figure 3.34. The *Use Original Location* button in figure 3.34 will set the location to the location the building currently is associated with as opposed to the user’s current location. Once the updates have been made, the user can press the *Update* button to perform the update. If no updates have been made, an error will be displayed to that effect as shown in figure 3.35. Otherwise the user will be returned to the updated originating current building view as shown in figure 3.22.

3.4.2 Deleting a building

A building can be deleted as a logged in user from either the building map view in figure 3.15 or the building list view in figure 3.16. From the building map view in figure 3.15, tapping the map icon for the building in question will show a new overlay view for the building as shown in figure 3.21. When the *Delete* button in figure 3.21 is tapped, the building will be deleted. From the building list view in figure 3.16, tapping the *Delete* button for the building in question will perform the deletion.
3.4.3 Updating a bathroom

A bathroom can be updated as a logged in user from either the bathroom amenities view in figure 3.26 or the bathroom reviews view in figure 3.27 by pressing the Update button in the top right of the screen. The user will be taken to the update bathroom view filled out with the current bathroom data filled in as shown in figure 3.36. Once the updates have been made, the user can press the Update button to perform the update. If no updates have been made, an error will be displayed to that effect as shown in figure 3.37. Otherwise the user will be returned to the updated originating bathroom amenities view from figure 3.22 or in figure 3.26 or bathroom reviews view in figure 3.27.

3.4.4 Deleting a bathroom

A bathroom can be deleted as a logged in user by tapping the Delete button for the bathroom in question from the current building view as shown in figure 3.22.
3.4.5 Updating an amenity

An amenity can be updated as a logged in user by tapping the Update button for the amenity in question from the bathroom amenities view as shown in figure 3.26. The user will be taken to the update amenity view with the current amenity data filled in as shown in figure 3.38. Once the updates have been made, the user can press the Update button to perform the update. If no updates have been made, an error will be displayed to that effect as shown in figure 3.39. Otherwise the user will be returned to the updated originating bathroom amenities view from figure 3.26.

3.4.6 Deleting an amenity

An amenity can be deleted as a logged in user by tapping the Delete button for the amenity in question from the bathroom amenities view as shown in figure 3.26.
3.4.7 Updating a review

A review can be updated as a logged in user by tapping the Update button for the amenity in question from the bathroom reviews view as shown in figure 3.26. The user will be taken to the update review view with the current review data filled in as shown in figure 3.40. Once the updates have been made, the user can press the Update button to perform the update. If no updates have been made, an error will be displayed to that effect as shown in figure 3.41. Otherwise the user will be returned to the updated originating bathroom amenities view from figure 3.26.

3.4.8 Deleting a review

A review can be deleted as a logged in user by tapping the Delete button for the review in question from the bathroom reviews view as shown in figure 3.27.
3.5 Find Bathroom Scenarios

3.5.1 Searching for bathrooms

The search bathrooms view provides an easy way to search for bathrooms and can be accessed via the WC Locator tab. Initially the user will be presented with the notes view shown in figure 3.42. The user can press the Search button at the top of figure 3.42 to navigate to the search view shown in figure 3.43. By default, search will be in a radius of 1 mile from the user's current location, nearest to farthest, and search on bathroom name. The user can search by pressing the Search button under the search bar as shown in figure 3.44.
3.5.2 Using search filters

Search filters can be accessed via the Filters button in the search bathrooms view in figure 3.43. The user will be taken to the apply filters view shown in figure 3.45. Filter sections can be viewed and hidden as shown between figures 3.45 and 3.46. A filter option will have a blue checkmark next to it if it is selected as shown in figure 3.47. The floors and ratings filters are both range filters and are shown in figure 3.48. The Apply Filters button sets the current set of filters to those used in the search bathrooms view and returns the user to the search bathrooms view as shown in figure 3.43. The Clear All button clears all current filters to the default search filter state.
3.5.3 Viewing search results

Bathroom search results can be viewed after performing a search as shown in figure 3.44. An individual bathroom search result can be viewed by pressing the View button for the search result in question. The user will be taken to the bathroom search results info view as shown in figure 3.49. The user can also view the bathroom's associated amenities and reviews by tapping the corresponding buttons as shown in figures 3.50 and 3.51.

3.6 Note Scenarios

3.6.1 Viewing existing notes

Notes are posts users can make about bathrooms separate from the bathrooms themselves. Notes can be viewed via the WC Locator tab. The notes list view is the initial view presented in the WC Locator tab as shown in figure 3.42. The notes list view is divided into sections: info notes and event notes. Info notes are general notes about anything. Event notes are specifically meant for event related bathroom information.
3.6.2 Adding a note

Notes can be added as a logged in user by tapping the *Add Note* button in the notes list view in figure 3.42. The user will then be taken to the add note view shown in figure 3.52. After filling out the fields in the add note view, the user can tap the *Add* button to create the new note as shown in figure 3.53. If any data is missing or incorrect in figure 3.53 an error will be displayed describing the error as shown in figure 3.54. Otherwise the user will be taken back to the updated originating building view from figure 3.42.
3.6.3 Updating a note

Notes can be updated as a logged in user by tapping the *Update* button for the note in question in the notes list view in figure 3.42. The user will then be taken to the update note view shown in figure 3.55. Once the updates have been made, the user can press the *Update* button to perform the update. If no updates have been made, an error will be displayed to that effect as shown in figure 3.56. Otherwise the user will be returned to the updated originating notes list view from figure 3.42.

3.6.4 Deleting a note

A Note can be deleted as a logged in user by tapping the *Delete* button for the note in question from the notes list view as shown in figure 3.42.
4 Reflection

Overall, development of this app went very well. I gained some good experience developing a distributed mobile app as would be typical of an industry mobile app project. I also got more experience with using Autolayout constraints to build my app’s UI and the Swift programming language, both of which are relatively new technologies. This project also gave me more experience with developing for new technologies as they were being released. Both the iPhone X and the release of Swift 4 required design changes I wouldn’t have had to consider otherwise.
While the app currently works as expected, the distributed nature of the app makes it such that updating the REST API later may break existing functionality. Creating a development version of the REST API will make future updates easier to perform. Porting the app to Android would also be worth doing to increase the number of potential app users. Optimizing the app more for smaller screen sizes will also aide in supporting more phones, which would be useful as a reference point for other mobile platforms like Android. I also still intend to release the app on the app store, which should take place in the near future.

While the app has plenty of features geared towards bathroom consumers, the app doesn’t have as much for the businesses that own the bathrooms. Allowing businesses to claim their bathrooms and check if the bathrooms are in urgent need of cleaning would make the app even more useful to everyone and expand the target audience.
6 Conclusion

Creating the PubWC review app was a good experience overall. I built a fully functional application to handle crowdsourced data. I was able to complete the minimum viable product and add some additional features to make the app even better. I also built the app in a scalable manner so it could be improved to support other platforms in future. Overall I’m proud of having completed this iOS application and I look forward to making more iOS applications in the future.
References


