Oakland Inconvenience Tracker

Software Requirements Specification

1.1

10/7/2018

Dylan Sporrer
David Rowan
Brandon Donahue
Adam Farabaugh
Software Engineering Team

Prepared for
University of Pittsburgh: CS 1530

Revision History

Date	Description	Author	Comments
9/26/2018	Beta Version 1.0	Dylan Sporrer	Initial Creation
10/2/2018	Beta Version 2.0	David Rowan	Added Classes and Class Diagram
10/7/2018	Release Version 1.0	Dylan Sporrer	Finished First Draft
10/8/2018	Release Version 1.1	David Rowan	Updated with DB Diagram

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Dylan Sporrer	Software Eng.	
	David Rowan	Software Eng.	
	Brian Donahue	Software Eng.	
	Adam Farabaugh	Software Eng.	

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to catalog the goals of the "Oakland Inconvenience Tracker" application. It will establish and elaborate upon a clear set of goals set by the development team by outlining the functional and non-functional requirements necessary to achieve these goals. It will also provide information regarding other areas of early-stage design, including use-cases and various UML diagrams.

1.2 Scope

The Oakland Inconvenience Tracker will aid users in maintaining awareness of unexpected issues in the Oakland area and how said issues may affect transit or access to campus facilities. It will provide users will a way to view inconveniences (irregular facility shut-downs, construction, etc.) that may hinder their commute across campus or otherwise alter their normal event schedule. The catalog of inconveniences will be user-base sourced, as users will have the ability to add new inconveniences which they encounter as well as details about their nature. As such, the application will allow for better overall navigation of the area by users in Oakland and help to prevent minor issues from expanding due to an unaware Oakland populace.

1.3 Definitions, Acronyms, and Abbreviations

Definitions

Inconvenience: An unexpected issue in the Oakland area of which the app should inform users.
Listing: One instance of an inconvenience and its associated information in the app's database.
Logged-In User: A user whose current session with the application has been associated with a single profile from a database of valid profiles
Administrative User: A user of the system whose associated profile has been granted special permissions and abilities beyond that of other users.

Acronyms

OIT: The Oakland Inconvenience Tracker, both the name of the development project as well as the final application

2. General Description

2.1 Product Perspective

The OIT is a stand-alone application which is intended to be developed as a self-contained product. It is being produced within the structure of the University of Pittsburgh CS 1530 course, but is not associated with any other products developed in the same course.

2.2 Product Functions

The OIT will carry out three primary functions. The first is to allow users to create inconvenience listings pertaining to the Oakland area. The second is to allow users to view inconvenience listings, both in tabular and map-overlay form, and filter the total catalog of said

listings by relevancy to them. The third is to implement a system by which irrelevant or obsolete inconvenience listings are automatically removed from the application's database.

2.3 Constraints, Assumptions and Dependencies

2.3.1 – Constraints
2.3.1.1 – The volume of listings and profiles are constrained by the database system utilized
2.3.2 - Dependencies
2.3.3.1 - The map overlay for inconveniences requires the use of the Google Maps API
2.3.3.2 – Amazon Web Services will be utilized to implement the online requirements

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 Software Interfaces

3.1.1.1 – Google Maps API
The live map system will be implemented using the Google Maps API. The Listing system will communicate location and inconvenience type information the maps API so as to create unique pins for each listing at the appropriate location. This map will then be visible and interactable on the primary (landing) page of the service.

3.2 Functional Requirements

3.2.1 User Profile System

3.2.1.1 Description
The user profile system will allow for differentiation between users who are logged in to a recorded profile and those who are not. This distinction will determine the extent of a user's privileges. This is a high-priority item.

3.2.1.2 Stimulus Sequences
The User Profile System will be required to act any time a user chooses to attempt to create, log in to, or log out of, a user profile.

3.2.1.3 Functional Requirements

3.2.1.3.1 - The system should allow users to enter a new unique email and password set to create an associated entry in the profile database

3.2.1.3.2 - The system should send a validation code to the email given during profile creation and expect this code to be entered by a user before the profile is created

3.2.1.3.3 - The system should verify that a given email and password combination uniquely identifies an existing entry in the profile database when a user attempts to log in

3.2.1.3.4 – The system should verify that the profile associated with the information given during log-in is not currently in use

3.2.1.3.5 - The system should communicate to other systems whether a user interacting with them is or is not currently logged in to a valid profile

3.2.1.3.6 – The system should communicate to other systems whether a user with a valid profile is an administrative user or not

3.2.1.3.7 – The system should allow a user to log-out of a profile and communicate that the profile is no longer in use to the profile database

3.2.2 Inconvenience Listing System

3.2.2.1 Description

The inconvenience listing system will allow for any user to view a catalog of listings and well as allow logged-in users to add new listings to this catalog. This system will also handle the approval rating of a given listing and the associated automatic actions. This is a vital and high priority item.

3.2.2.2 Stimulus Sequences

The inconvenience listing system will be required to act whenever a user chooses to view the current catalog of listings or add to this catalog.

3.2.2.3 Functional Requirements

3.2.2.3.1 - The system will allow any user to view all existing listings in a catalog format

3.2.2.3.2 – The system will allow any user to filter this catalog by a set of tags selected from the tag system's dictionary of tags

3.2.2.3.3 - The system will allow any user to view the details of a particular listing

3.2.2.3.4 – The system will allow logged-in users to create new listings where a listing is comprised of the following:

- Inconvenience Type (Construction, Room-In-Use, Out-of-Order, Pothole, or Crowded)
- Location
- Optional Detailed Description
- Set of Associated Tags
- Optional Image
- Time of Fist Posting
- Time of Most Recent Vote Update
- Number of Positive Votes
- Number of Negative Votes

3.2.2.3.5 – The system will allow logged-in users to either increase or decrease the positive or negative vote total by one for any posting a single time

3.2.2.3.6 – The system will allow administrative users to remove any listing from the catalog

3.2.2.3.7 – The system will maintain a relevancy value for each listing based on type of inconvenience, time of posting, number of positive votes, and number of negative votes

3.2.2.3.8 – The system will automatically remove from the catalog any listing which has a negative relevancy value

3.2.2.3.9 – The system will communicate any updates to the catalog to the live map system and tag system

3.2.3 Tag Update System

3.2.3.1 Description

The tag update system maintains a set of labels which a listing can be associated with. It also allows for users to select a set of tags they wish to associate with their profile and updates that profile when an associated tag is updated. These updates can come in the form of a new listing with the given tag or the removal of a listing which had the given tag.

3.2.3.2 Stimulus Sequences

The tag system is required to act when an administrator wishes to edit the dictionary of available tags, when a user wishes to update their set of monitored tags, or anytime a listing which has one or more tags is added or removed.

3.2.3.3 Functional Requirements

3.2.3.3.1 – The system will maintain a dictionary of valid tags which represent buildings, streets, facilities, or classes which may be affected by an on-campus inconvenience

3.2.3.3.2 – The system will allow a user to associate one or more tags from the dictionary with an account which they are logged in to

3.2.3.3.3 – Upon receiving notification from the Listing System that a new listing has been created with a set of tags, the system will notify all profiles which have been associated with any of the listing's tags

3.2.3.3.4 – Upon receiving notification from the Listing System that a listing has been removed which contained a set of tags, the system will notify all profiles which have been associated with any of the listing's tags

3.2.3.3.5 – The system will allow any logged-in user to view the dictionary of tags

3.2.3.3.6 – The system will allow administrative users to add to or remove from the dictionary of valid tags

3.2.4 Live Map Display

3.2.4.1 Description

The live map system translates the information maintained by the Listing and Tag systems to a visual form overlaid on a pre-existing Google Maps representation of Oakland. It places the listings managed by the catalog according to user-provided locations and allows users to inspect the associated markers as they would inspect a listing in the catalog.

3.2.4.2 Stimulus Sequences

The live map system will be required to act when either a user chooses to view the map it generates or when another system notifies it of changes to the listing catalog.

3.2.4.3 Functional Requirements

3.2.4.3.1 – The system will allow any user to view the listing catalog overlaid on a Google Maps representation of the Oakland area

3.2.4.3.2 – The system will utilize a different visual marker for each type of inconvenience

3.2.4.3.3 – The system will allow any user to filter this view of the listings by a set of tags

3.2.4.3.4 – The system will allow any user to view the details of a particular listing by selecting its visual representation on the map

3.2.4.3.5 – The system will automatically update the map when receiving notifications from the inconvenience listing system

3.3 Use Cases

3.3.1 Login Feature Use Cases

Use Case ID:	UC-3.3.1.1		
Use Case Name:	Create Profile		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	9/13/2018	Last Revision Date:	9/13/2018
Actors:	Primary: Profile-less User Secondary: E-Mail Service		
Description:	This case is for the event that a user new to the system wishes to create a profile, allowing them to access the full feature set		
Trigger:	A user not logged in to any existing profile chooses to create one		
Preconditions:	1. User is not logged in to any existing profile		
Postconditions:	Success Guarantees: 1. User login info is stored securely in database 2. User is logged in to new profile		
Normal Flow:	1. User selects “Create New Profile” 2. System prompts user to enter email address, password, and password confirmation 3. System validates no profile exists with given email address 4. System validates password matches restrictions 5. System sends profile confirmation message to provided email address		

	6. System accepts response from confirmation page 7 User is logged in to new profile
Alternative Flows: **[Alternative Flow 1 – Profile with Given Email Exists]**	3a. In step 3 of the normal flow, if a profile exists with the given email 1. System will prompt user to login to existing profile or attempt to create new profile 2. User selects existing profile login 3. User enters UC-1.2.2 3b. In step 3 of the normal flow, if a profile exists with the given email 1. System will prompt user to login to existing profile or attempt to create new profile 2. User selects create new profile 3. Use Case resumes on step 2 of normal flow
[Alternative Flow 2 – New Password Invalid]	4a. In step 4 of the normal flow, if the password constraints are not met 1. System informs user of issue 2. Use case resumes on step 2 of normal flow
Exceptions:	5a. In step 4 of the normal flow, if the user enters and invalid email 1. Customer informed that email is non-existent 2. Use Case resumes on step 2 of normal flow
Includes:	N/A
Frequency of Use:	On Demand; High volume early in life of product, low volume later
Special Requirements:	1. Ability to store login information securely 2. Ability to interact with common email platforms
Assumptions:	1. User has existing email address and is capable of utilizing it 2. User is capable of creating good-strength password
Notes and Issues:	1. Maximum password length and contents TBD

Use Case ID:	UC-3.3.1.2		
Use Case Name:	Access Profile		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	9/13/2018	Last Revision Date:	9/17/2018
Actors:	Primary: Any User		
Description:	This case is for the event that a user wishes to access an existing user profile.		
Trigger:	A user not logged in to any existing profile chooses to log into one.		
Preconditions:	1. User is not logged in to any existing profile. 2. At least one user profile exists in the database which is not in use		
Postconditions:	Success Guarantees 1. User is logged in to existing profile 2. Login info remains secure in database		
Normal Flow:	1. User selects “Log In” 2. System prompts user to enter email and associated password 3. System validates that email and password uniquely identify an existing and not-in-use user profile 4. User gains access to existing profile		
Alternative Flows: **[Alternative Flow 1 – No Existing Profile Found]**	3a. In step 3 of the normal flow, if the email and password do not uniquely identify an existing user profile 1. System informs user of inability to locate expected profile 2. Provided password field is cleared 3. Resume normal flow at step 2		
Alternative Flows: **[Alternative Flow 2 – Specified Profile in Use]**	3b. In step 3 of the normal flow, if the uniquely identified profile is currently in use 1. System informs user profile is in use 2. Provided login fields are cleared		

	3. Resume normal flow at step 2
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	On Demand
Special Requirements:	1. Ability to store login data securely
Assumptions:	1. User knows the login information to a pre-existing user profile
Notes and Issues:	N/A

Use Case ID:	UC-3.3.1.3		
Use Case Name:	Recover Password		
Created By:	David Rowan	Last Updated By:	Dylan Sporrer
Date Created:	9/25/18	Last Revision Date:	9/29/18
Actors:	Primary: Any User		
Description:	User is trying to access their account and has forgotten their password.		
Trigger:	User clicks “Forgot Password”		
Preconditions:	1. User is not logged in		
Postconditions:	2. User resets their password in the system		
Normal Flow:	1. User clicks “Forgot Password” 2. User enters their email 3. System validates that profile with given email exists 4. System generates new password 5. New password is sent to user’s email 6. System returns to login screen		
Alternative Flows:	3a. In step 3 of the normal flow, if the entered email does not identify a profile in the database 1. The system informs user of error 2. Resume normal flow at step 6		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	On Demand		
Special Requirements:	1. Email Server Be available		
Assumptions:	1. Email Server available		
Notes and Issues:	1. Exact format of generated passwords TBD		

3.3.2 Inconvenience Listing Feature Use Cases

Use Case ID:	UC-3.3.2.1		
Use Case Name:	Post Listing		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	9/19/2018	Last Revision Date:	9/24/2018
Actors:	Primary: Signed in User		
Description:	This case occurs when a user which is already signed in to a profile wishes to add a new inconvenience listing to the catalog of inconveniences		
Trigger:	User chooses to create new listing		
Preconditions:	1. User is logged in to existing profile 2. Listing catalog has space available for new listing		
Postconditions:	Success Guarantees: 1. New listing is added to catalog		
Normal Flow:	1. User selects “Create Listing” 2. System prompts user to select inconvenience type, location (by selecting on map screen), and any affected tags		

	3. System prompts user for additional details in form of text box 4. System prompts user to upload optional image of inconvenience 5. User chooses to upload image 6. User selects image from local file system 7. Listing is added to catalog 8. Tag system is notified of new listing with given tags 9. Map system is updated with the location of listing
Alternative Flows: **[Alternative Flow 1 - User Foregoes Uploading Image]**	5a. In step 5 of the normal flow, if user chooses not to upload an image 1. System provides default image for listing 2. Resume normal flow at step 7
Exceptions:	5b. In step 5 of the normal flow, if user chooses file of unsupported type 1. System informs user of invalid type 2. Resume normal flow at step 4
Includes:	N/A
Frequency of Use:	On Demand
Special Requirements:	1. Ability to insert new listings into existing database
Assumptions:	2. User can locate own location on provided map
Notes and Issues:	1. Valid image file formats TBD

Use Case ID:	UC-3.3.2.2		
Use Case Name:	View Listing Catalog		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	10/5/2018	Last Revision Date:	10/5/2018
Actors:	Primary: Any User		
Description:	This case occurs when a user wishes to view the existing inconvenience listings in a catalog format rather than on the live map		

Use Case ID:	UC-3.3.2.3		
Use Case Name:	View Details of Listing		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	10/5/2018	Last Revision Date:	10/5/2018
Actors:	Primary: Any User		
Description:	This case occurs when a user wishes to view the details (time of posting, description, etc.) of a specific listing found in the catalog view.		

Use Case ID:	UC-3.3.2.4		
Use Case Name:	Search Catalog by Tag		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	9/24/2018	Last Revision Date:	9/24/2018
Actors:	Primary: Any User		
Description:	This case occurs when a user chooses to view the catalog of inconveniences without the use of the map overlay and then curate this list by a set of tags.		
Trigger:	User chooses to search tabular catalog by set of tags		
Preconditions:	1. Catalog contains at least one existing listing 2. Tag dictionary contains at least one valid tag 3. User has access to tabular catalog of listings		
Postconditions:	Success Guarantees: 1. Catalog only displays listings with matching tag(s)		
Normal Flow:	1. User selects “Filter by Tag” 2. System prompts user to select any number of tags from dictionary		

	3. System prompts user to select “all” or “any” option for tag set 4. System searches catalog database for matching listings 5. System displays new catalog containing only the listings found by the previous tag search
Alternative Flows: **[Alternative Flow 1 – No Matching Listings]**	3a. In step 3 of the normal flow, if no listings can be found which adhere to the search settings 1. System informs user of failure to find 2. System returns user to unfiltered catalog 3. Resume normal flow at step 1
Exceptions:	3b. In step 3 of the normal flow, if the listing database cannot be located or is empty 1. System informs user of issue 2. User returned to empty catalog view
Includes:	UC 2.2.2: User Views Catalog of Listings
Frequency of Use:	On Demand
Special Requirements:	1. Ability to search database by tags
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	UC-3.3.2.5		
Use Case Name:	Admin Remove Listing		
Created By:	Brandon Donahue	Last Updated By:	Brandon Donahue
Date Created:	9/23/18	Last Revision Date:	9/23/2018
Actors:	Primary: Admin		
Description:	This case is for the event that an admin wants to remove a listing from the map manually.		
Trigger:	Admin selects a listing from the map and chooses the “remove” option, only available to admins.		
Preconditions:	1. Admin is logged into admin account with special privileges,		
Postconditions:	Success Guarantees: 1. Selected listing is removed from the map and the database.		
Normal Flow:	1. Admin selects a listing on the map 2. Admin selects the “remove” option on the listing 3. System asks admin to confirm the removal of a listing. 4. Upon confirmation, the listing is removed from the map, and the database is updated accordingly.		
Alternative Flows: **[Alternative Flow 1 – Admin does not confirm the removal of a listing]**	3a. In step 3 of the normal flow, if the admin does not confirm they wish to remove a listing 1. Exit the use case		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Low Volume, as most listings should remove themselves		
Special Requirements:	1. Ability to identify an account as an admin.		
Assumptions:	1. Admin responsibly removes listings, and lets the system run on its own if there are no unusual behaviors.		
Notes and Issues:			

3.3.3 Tag System Use Cases

Use Case ID:	UC-3.3.3.1		
Use Case Name:	Add Tag to Listing		
Created By:	David Rowan	Last Updated By:	Dylan Sporrer
Date Created:	9/25/18	Last Revision Date:	9/28/18
Actors:	Primary: Any User		
Description:	When a user is creating a listing for the map they have the option of adding a tag to the map to further describe the event. I.E. A crowd tag to show a crowd or people.		
Trigger:	User selects add tag when creating a listing		
Preconditions:	1. User must be logged in. 2. User must be creating a listing		
Postconditions:	1.The new listing will have a tag associated with 2. An Icon related to the icon will be shown as well. This will alter the image on the map based on the tag.		
Normal Flow:	1. User clicks “Add Tag” 2. User scrolls through a list and selects icon 3. Icon is then applied to their listing before creation		
Alternative Flows:			
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	On Demand		
Special Requirements:	N/A		
Assumptions:	1.Icons are available 2.User is allowed to create listings		
Notes and Issues:			

Use Case ID:	UC-3.3.3.2		
Use Case Name:	View Dictionary of Tags		
Created By:	Adam Farabaugh	Last Updated By:	
Date Created:	9/24/18	Last Revision Date:	
Actors:	Primary: Any User		
Description:	This case is for the event in which the user wishes to view all possible tags they can attach to a pin		
Trigger:	A user chooses the “view tags” option		
Preconditions:			
Postconditions:			
Normal Flow:	1. User Selects the “tags” menu 2. User Clicks on “view all tags” 3. The tags appear and the user can read them.		
Alternative Flows:	1. There is a connection error and a message is displayed		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	On Demand		
Special Requirements:	1. Ability to retrieve data from the database		
Assumptions:	1. The database is connected and running		
Notes and Issues:	@Preconditions – user not need to be logged in to see possible pins?		

Use Case ID:	UC-3.3.3.3		
Use Case Name:	Set Tags to Receive Updates About		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	10/5/2018	Last Revision Date:	10/5/2018
Actors:	Primary: Logged-In User		
Description:	This case occurs when a user with a valid profile chooses to select the tag or tags they wish to receive updates regarding. (either the addition or removal of listings with the given tag)		

Use Case ID:	UC-3.3.3.4		
Use Case Name:	Admin Tag Dictionary Edit		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	10/5/2018	Last Revision Date:	10/5/2018
Actors:	Primary: Administrative User		
Description:	This case occurs when an administrator chooses to add or remove tags from the dictionary of possible tags for listings, pins, and searches.		

3.3.4 Live Map Feature Use Cases

Use Case ID:	UC-3.3.4.1		
Use Case Name:	View Live Map		
Created By:	Dylan Sporrer	Last Updated By:	Dylan Sporrer
Date Created:	10/5/2018	Last Revision Date:	10/5/2018
Actors:	Primary: Any User		
Description:	This case occurs when a user wishes to view the live map which reflects the location and type of all inconveniences listings.		

Use Case ID:	UC-3.3.4.2		
Use Case Name:	Search Map by Tag(s)		
Created By:	Brandon Donahue	Last Updated By:	Brandon Donahue
Date Created:	9/24/2018	Last Revision Date:	9/24/2018
Actors:	Primary: User		
Description:	This case is for the event that a user wants to search the map for one or more of the tags the system provides to the user to describe an occurrence.		
Trigger:	A user interacts with the search feature on the map.		
Preconditions:	1. User is on the map interface		
Postconditions:	Success Guarantees: 1. User is given a list of occurrences marked with the tags they chose		
Normal Flow:	1. User selects “search” function on the occurrence map 2. User begins selecting tags from a predetermined list of tags 3. When the user has selected all the tags they wish, they confirm their search 4. User selects whether the tag set is “all” or “any” 5. The user is then shown all of the occurrences with their given tags in a certain area around the user.		
Alternative Flows: **[Alternative Flow 1 – User did not select any tags]**	3a. In step 3 of the normal flow, if user confirms without selecting any tags 1. System will tell the user they must select at least one tag 2. Use Case resumes on step 2 of normal flow.		
[Alternative Flow 2 – No	4a. In step 4 of the normal flow, if the database does not contain any		

occurrences found with specified tags]	occurrences with the specified tags. 1. System informs user there are no occurrences matching their search criteria. 2. Use case terminates
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	High Volume throughout the lifetime of the product
Special Requirements:	1. Ability to interface with the database frequently
Assumptions:	
Notes and Issues:	1. List of tags TBD 2. Radius to search TBD

Use Case ID:	UC-3.3.4.3		
Use Case Name:	Add Pin to Map		
Created By:	David Rowan	Last Updated By:	
Date Created:	9/23/18	Last Revision Date:	
Actors:	Primary: Any User		
Description:	This case is for the event in which the user adds a pin to the map to list a occurrence in Oakland.		
Trigger:	A user chooses the add pin action		
Preconditions:	1.User is logged into the system 2. The user is allowed to post things to the map i.e. no permissions revoked		
Postconditions:	1. Pin is added to map 2. Pin location is added into the database 3. Other users can see the pin		
Normal Flow:	1. User Selects “Add Pin” on map 2. User the fills in the create a pin boxes(Location, Occurrence, etc) 3. User clicks “Create” and the pin is added to the map.		
Alternative Flows:	1. In Step 3 an error message appears because a the pin has already been added to that location by another user. 2. In Step 3 the user enters a location outside the bounds of the program.		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	On Demand		
Special Requirements:	1. Ability to store data and retrieve it 2. Access to google maps api		
Assumptions:	1. Google maps is working 2. The database is connected and running		
Notes and Issues:			

Use Case ID:	UC-3.3.4.4		
Use Case Name:	View Details of Pin		
Created By:	Adam Farabaugh	Last Updated By:	
Date Created:	9/24/18	Last Revision Date:	
Actors:	Primary: User		
Description:	This case is for the event in which the user views a pin on the map to see the details of the occurrence.		
Trigger:	A user chooses “view details” pin action.		

Preconditions:	
Postconditions:	1. The user is presented with specific details of the pin they selected
Normal Flow:	1. User Selects View Map 2. User Clicks on a pin on the map 3. The details appear and the user can read them.
Alternative Flows:	1. There is a connection error and a message is displayed
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	On Demand
Special Requirements:	1. Ability to retrieve data from the database 2. Access to google maps api
Assumptions:	1. Google maps is working 2. The database is connected and running
Notes and Issues:	@Preconditions – user not need to be logged in to see details?

Use Case ID:	UC-3.3.4.5		
Use Case Name:	Upvote/Downvote Pin		
Created By:	David Rowan	Last Updated By:	
Date Created:	9/23/18	Last Revision Date:	
Actors:	Primary: Any User		
Description:	This case is the case where a user can upvote/downvote an existing pin on the map to allows other users to see how helpful the pin is.		
Trigger:	A user upvotes a pin		
Preconditions:	1. User is logged into the system 2. The user is allowed to post things to the map i.e. no permissions revoked		
Postconditions:	1. Pin has updated count		
Normal Flow:	1. User selects either “Vote Up” or “Vote Down” on pin 2. System confirms that user has permissions to vote 3. System updates associated listing’s vote values 4. User is returned to map view		
Alternative Flows:	2a. In Step 2 of the normal flow, if user has revoked permission 1. System informs user of inability to affect vote values 2. Resume normal flow at step 4		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	On Demand		
Special Requirements:	1. Ability to store data and retrieve it 2. Access to google maps api		
Assumptions:	1. Google maps is working 2. The database is connected and running		
Notes and Issues:			

3.4 Classes / Objects

3.4.1 Landing Page

3.4.1.1 Attributes
- 3.4.1.1.1 - List mapCoordinate
- 3.4.1.1.2 - LoginInfo loginInfo

3.4.1.2 Methods
- 3.4.1.2.1 - Zoom()
- 3.4.1.2.2 - AddListing()
- 3.4.1.2.3 - UpVoteListing()
- 3.4.1.2.4 - DownVoteListing()
- 3.4.1.2.5 - BanUser()
- 3.4.1.2.6 - ViewLogs()
- 3.4.1.2.7 - RemoveListing()

3.4.2 Login Controller

3.4.2.1 Attributes
- 3.4.2.1.1 - String Username
- 3.4.2.1.2 - String Password

3.4.2.2 Methods
- 3.4.2.2.1 - returnAuthorizaed()
- 3.4.2.2.2 - returnPoints()
- 3.4.2.2.3 - resetPassword()

3.4.3 Admin Controller

3.4.3.1 Attributes
- 3.4.3.1.1 - string Username
- 3.4.3.1.2 - string Password
- 3.4.3.1.3 - string AdminLevel

3.4.3.2 Methods
- 3.4.3.2.1 - removeListing()
- 3.4.3.2.2 - viewOwnerListing()
- 3.4.3.2.3 - banUser()
- 3.4.3.2.4 - addAdmin()
- 3.4.3.2.5 - removeAdmin()

3.4.4 Map Info

3.4.4.1 Attributes
- 3.4.1.1.1 - List Coordinates
- 3.4.4.1.2 - String apikey
- 3.4.4.1.3 - Tuple currentLocation
- 3.4.4.1.4 - Tuple loginInfo

3.4.4.2 Methods
- 3.4.4.2.1 - returnListings()
- 3.4.4.2.2 - returnPastListings()
- 3.4.4.2.3 - addListing()
- 3.4.4.2.4 - addVote()

3.4.5 DB Context

3.4.5.1 Attributes
- 3.4.5.1.1 - String UserName
- 3.4.5.1.2 - String ip
- 3.4.5.1.3 - String AccessLevel

3.4.5.2 Methods
- 3.4.5.2.1 - returnListing()
- 3.4.5.2.2 - returnLogs()
- 3.4.5.2.3 - returnUsers()

3.4.6 Google Controller

3.4.6.1 Attributes
- 3.4.6.1.1 - String apiKey

3.4.6.2 Methods
- 3.4.6.2.1 - returnRequest()

3.4.7 Listing Class

3.4.7.1 Attributes
- 3.4.7.1.1 - Int Xcord
- 3.4.7.1.2 - Int YCord
- 3.4.7.1.3 - Int IconNum
- 3.4.7.1.4 - Int Description
- 3.4.7.1.5 - Int UpVotes
- 3.4.7.1.6 - Int DownVotes
- 3.4.7.1.7 - String Creator

3.4.7.2 Methods
- 3.4.7.2.1 - getDownvotes()
- 3.4.7.2.2 - setDownvotes()
- 3.4.7.2.3 - getXcord()
- 3.4.7.2.4 - setXcord()
- 3.4.7.2.5 - getYcord()
- 3.4.7.2.6 - setYcord()
- 3.4.7.2.7 - getIconNum()
- 3.4.7.2.8 - setIconNum()
- 3.4.7.2.9 - getDescription()
- 3.4.7.2.10 - setDescription()
- 3.4.7.2.11 - getUpvotes()
- 3.4.7.2.12 - setUpvotes()
- 3.4.7.2.13 - getCreator()
- 3.4.7.2.14 - setCreator()

3.4.8 Program Startup

3.4.8.1 Methods
- 3.4.8.1.1 - startProgram()
- 3.4.8.2.1 - configure()
- 3.4.8.3.1 - returnAppSettings()

3.5 Non-Functional Requirements

3.5.1 Performance

3.5.1.1 – The delay between a user request of Inconvenience Listing details and the retrieval of these details will not exceed 2 seconds
3.5.1.2 – The delay between a user listing submission and the addition of this listing to the live map overlay will not exceed 5 seconds

3.5.2 Reliability

3.5.2.1 – The application will have downtime not exceeding 1 hour per week

3.5.3 Availability

3.5.3.1 – The application will at minimum be available to the total Pitt student populace

3.5.4 Security

3.5.4.1 – Email account information associated with user profiles will remain inaccessible to all users
3.5.4.2 – Password information associated with user profiles will not be stored in the profile database in plaintext form

3.5.5 Maintainability

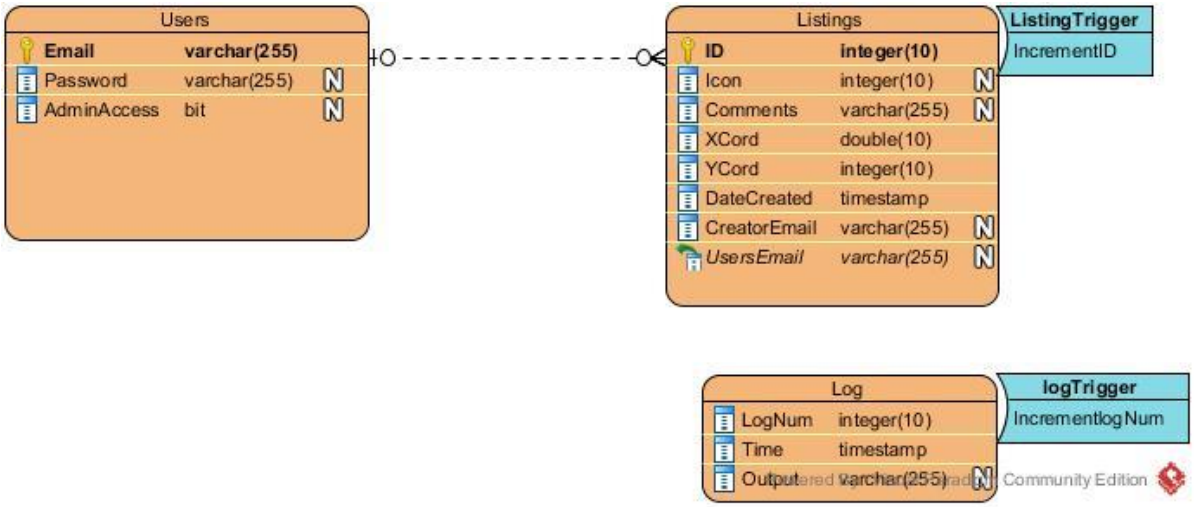
3.5.5.1 – Any Amazon Web Services updates will need to be incorporated to maintain the online aspects of the system

3.5.6 Portability

3.5.6.1 – The application will be transferrable across common browsers (Chrome, Firefox, IE)

3.6 Logical Database Requirements

3.6.1 – The database of listings, tags, and user profiles will remain stable indefinitely
3.6.3 – Diagram of Db tables below



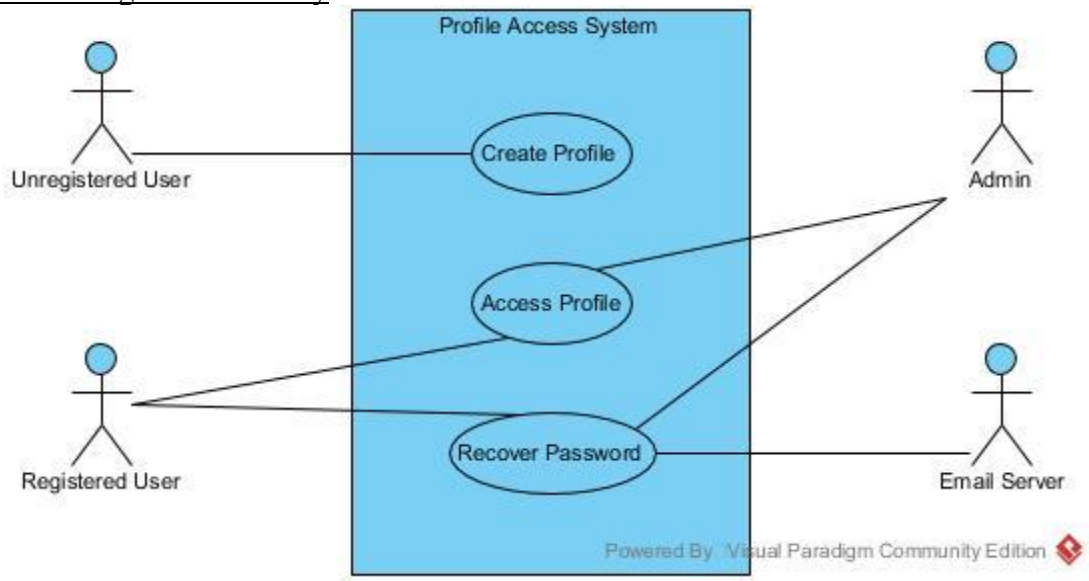
3.7 Other Requirements

3.7.1 – The Google Maps API is not applied without proper licensing or use of the developer version

4. Analysis Models

4.1 Use Case Diagrams

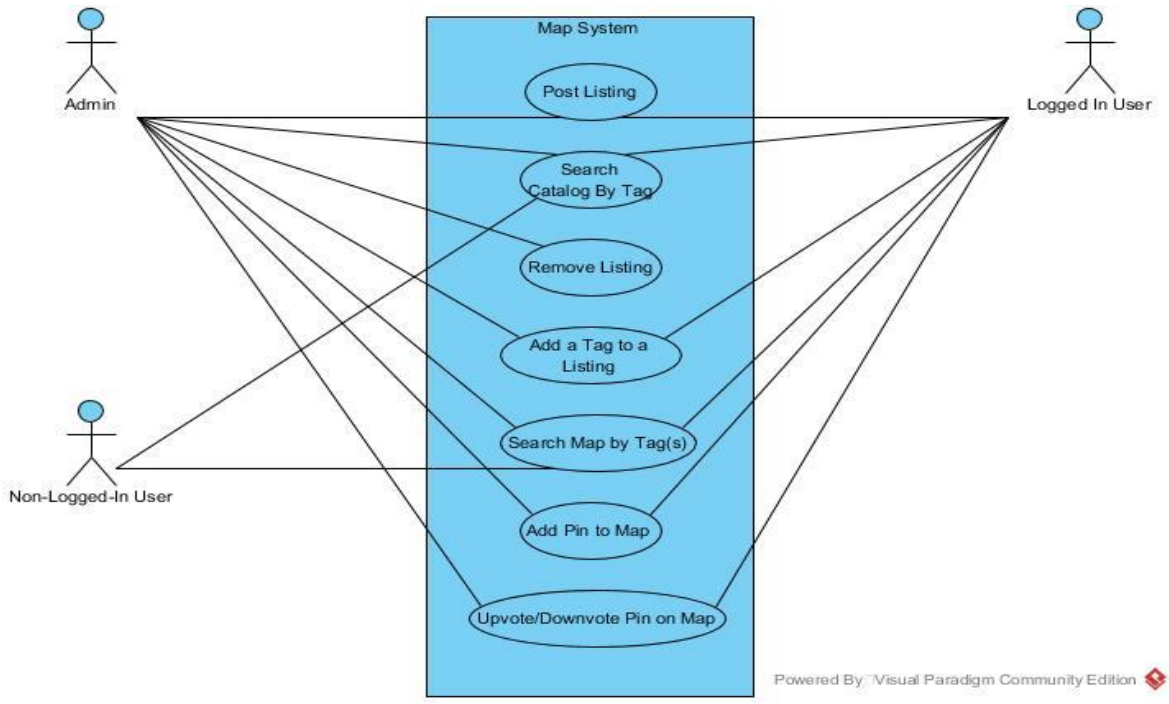
4.1.1 – Login and Security



The above diagram displays the login feature, separating the user actors into sub-groups of unregistered user, registered user, and administrative user. The email server does not innate the password recovery use-case but is expected to act after the use has started the process.

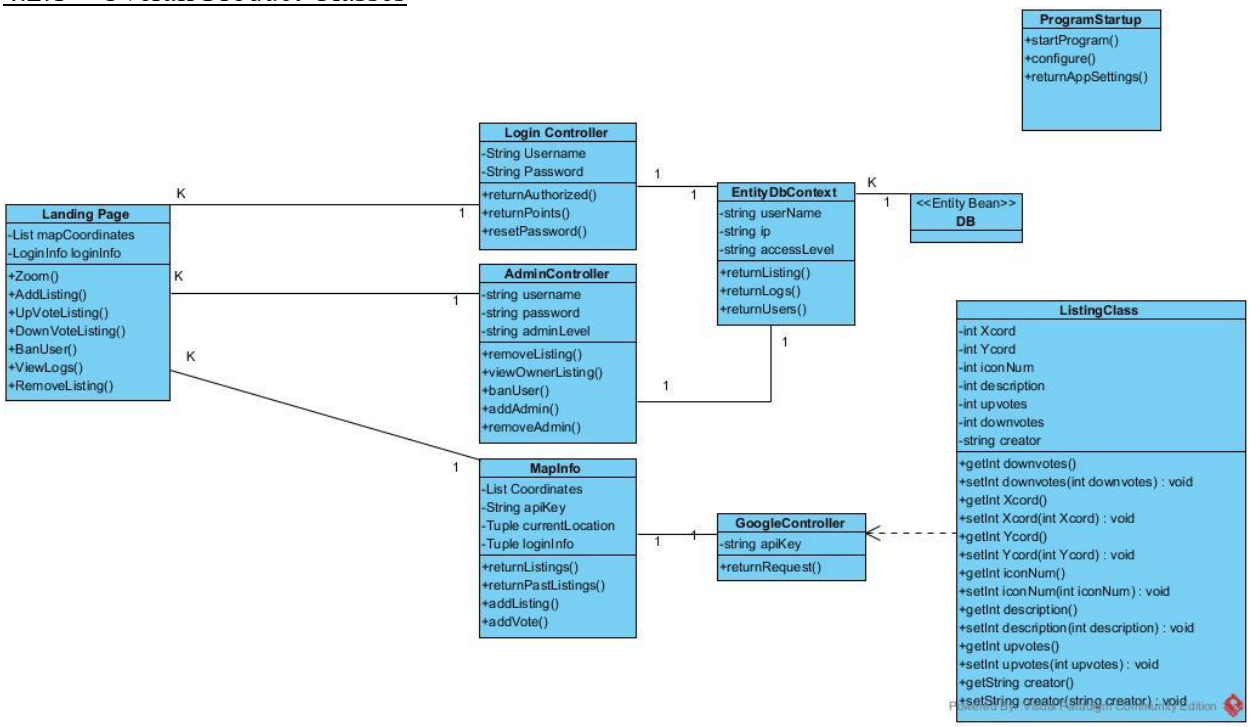
4.1.2 – Primary Service

The following diagram depicts the use cases which make up the interaction with the inconvenience listing and live map systems. Again the user actor is broken up into three different users with varying sets of permissions: a user not currently associate with a profile, a user associated with a profile, and a user associated with an admin profile.



4.2 Class Diagrams

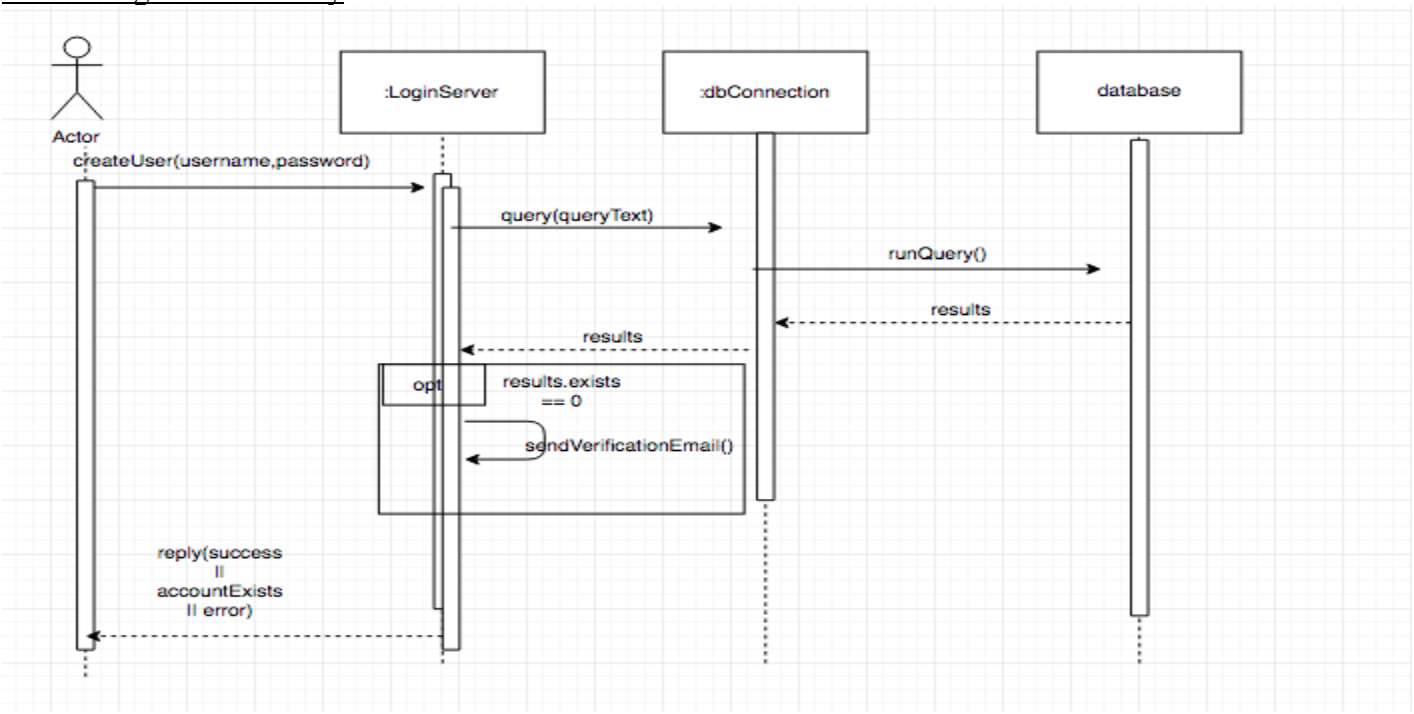
4.2.1 – Overall Product Classes

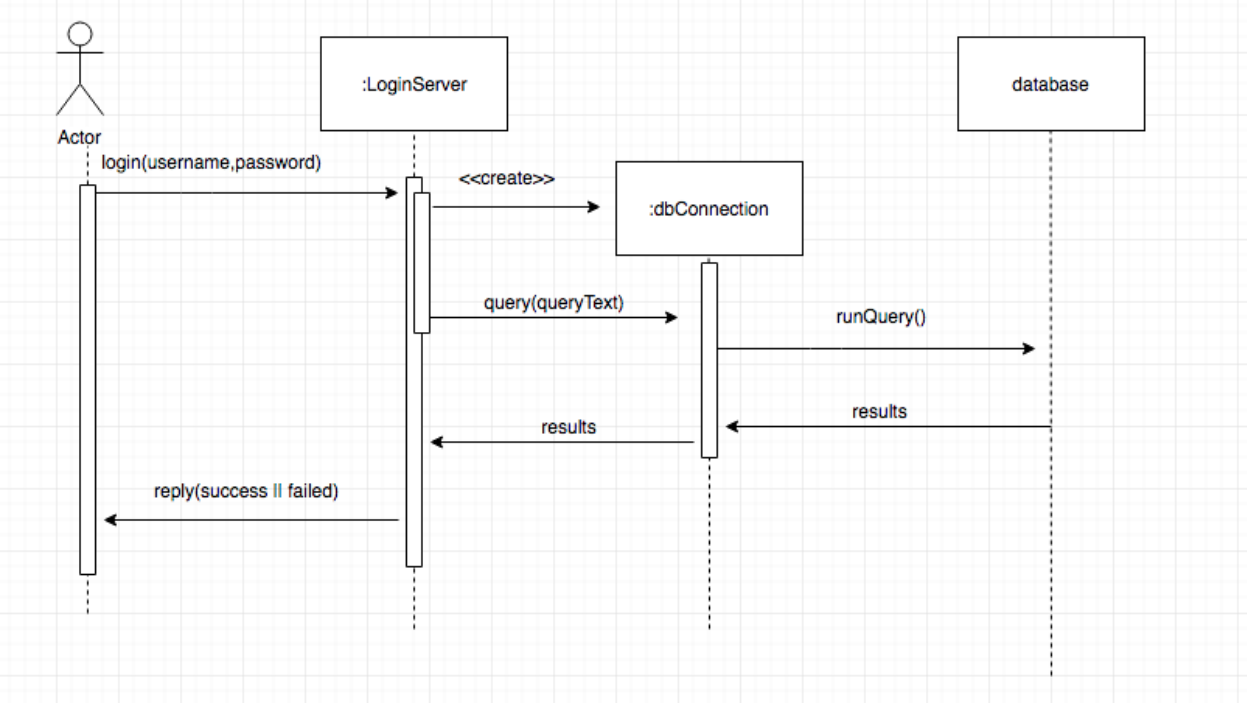


The above class diagram depicts the overall organization of the analysis classes in the system, with a landing page class acting as the sole point of interaction with the rest of the system.

4.3 Sequence Diagrams

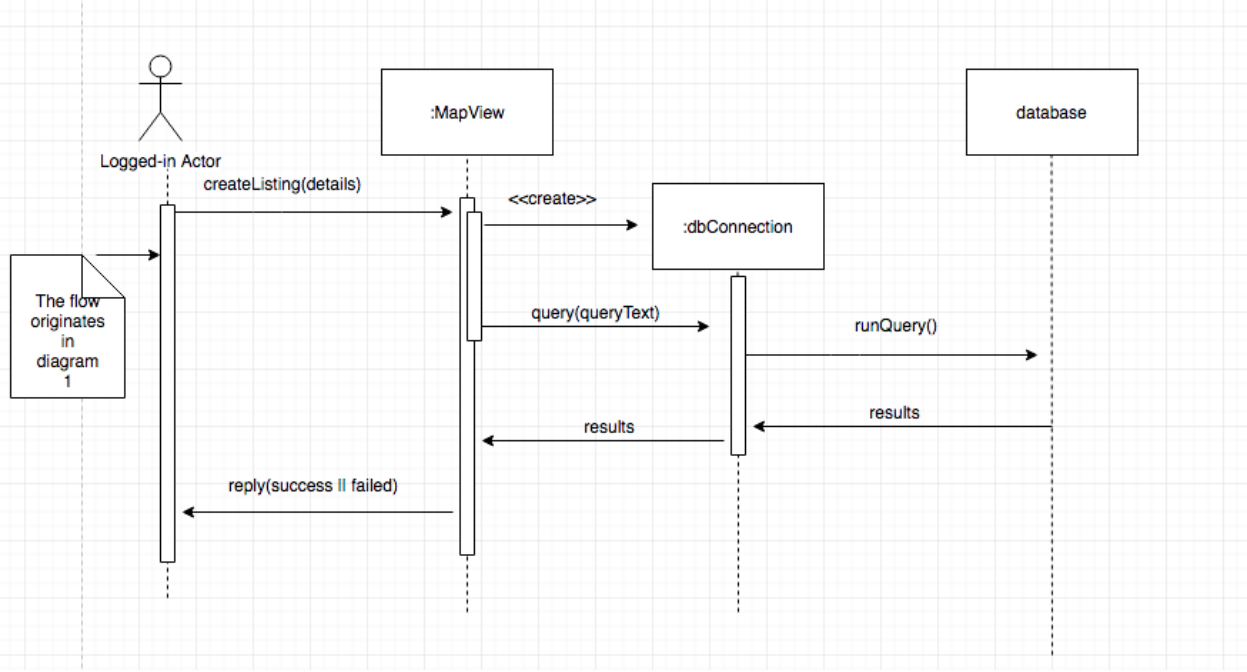
4.3.1 – Login and Security





The above two diagrams represent the profile creation and login services respectively. In both cases, a user's information entry prompts one or more queries to the database.

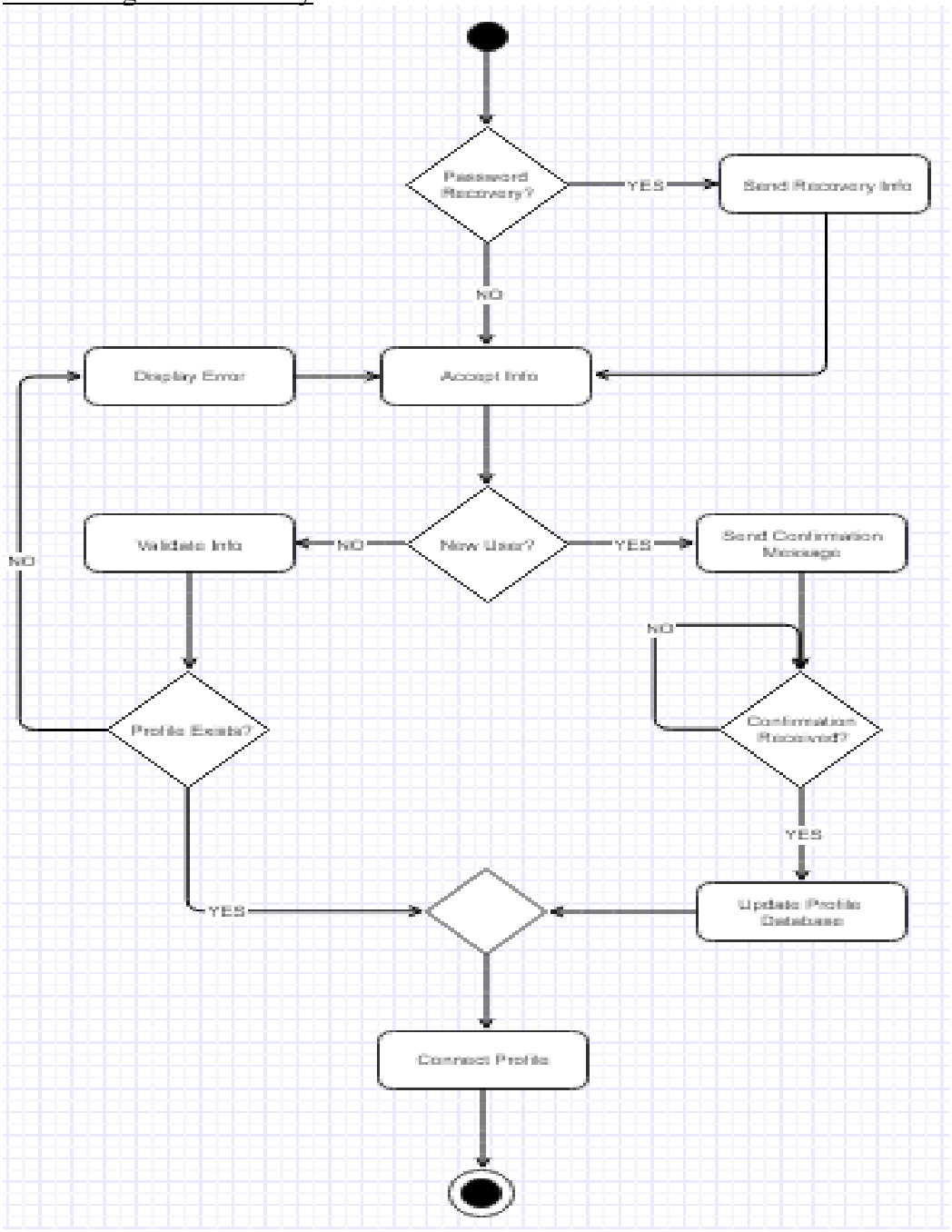
4.3.2 – Primary Service



The above diagram depicts a user creating a new listing through the map view. Once they submit an occurrence and fill out the details, a request will be sent and a connection to the database will be established. This query will insert a record into the database that has the details of the occurrence that has been observed by the user. The user will then be notified of a success or failure of this request.

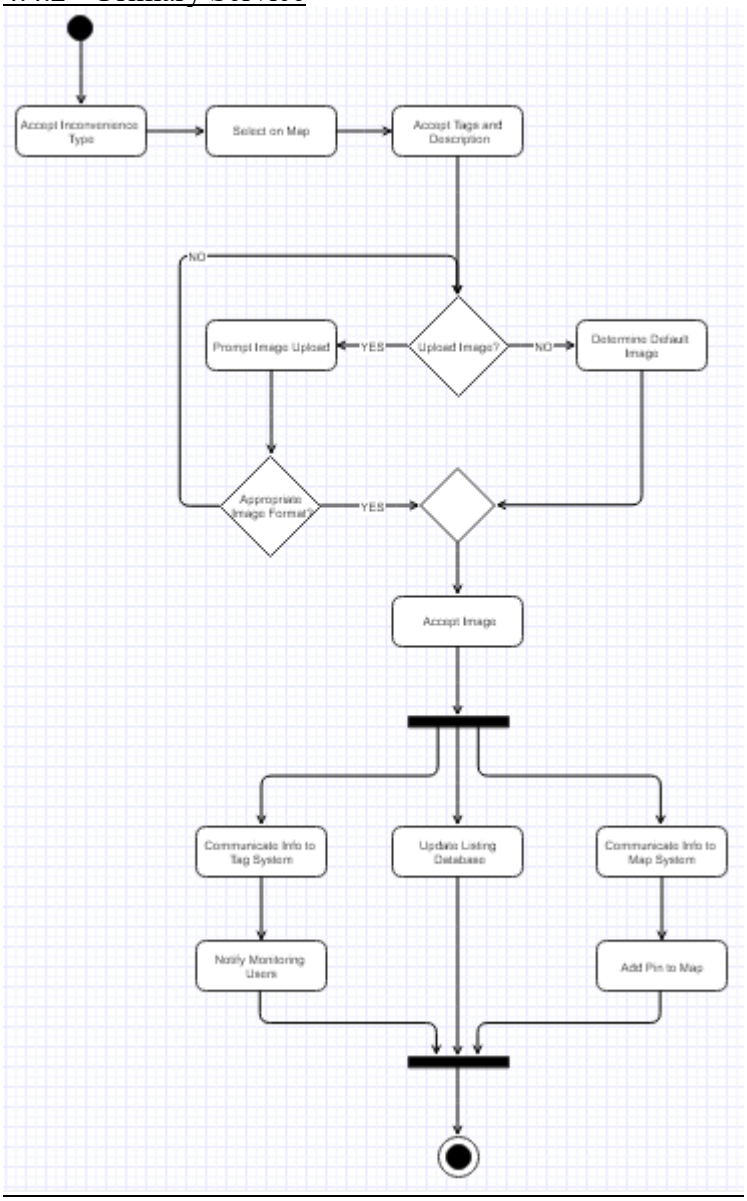
4.4 Activity Diagrams

4.4.1 – Login and Security



The above diagram depicts the workflow of the security function of the system, which encapsulates both the process of login for a pre-existing user and the process of profile creation for a new user. This workflow corresponds to UC 3.3.1.1 – UC 3.3.1.3.

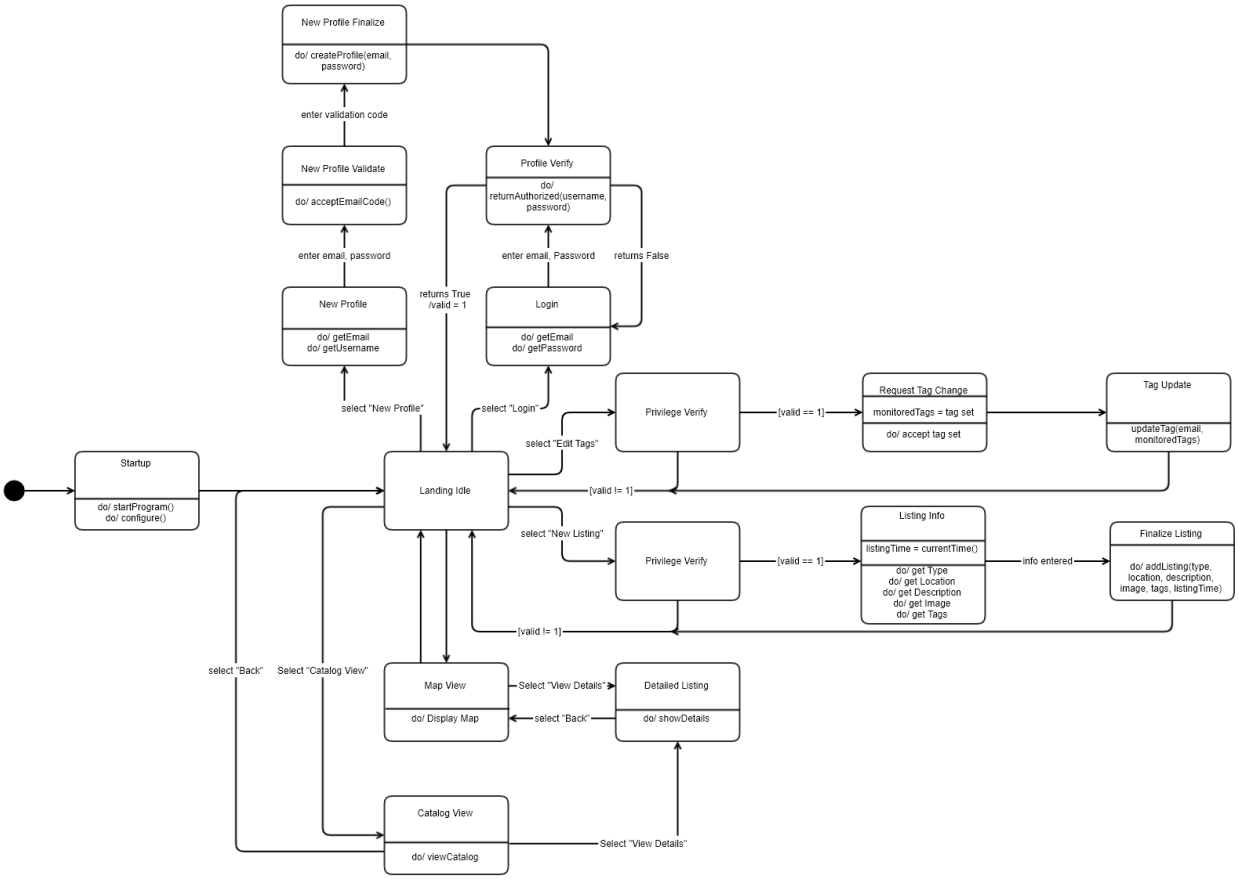
4.4.2 – Primary Service



The above diagram depicts the workflow of the primary function of the system, namely the creation of new inconvenience listings and the communication of this information to the map and tag system. The requirements for this workflow are found under section 3.2.2.3.

4.5 State-Transition Diagrams (STD)

4.5.1 – Overall Product States



The above diagram represents the network of states the entire system navigates during normal operation. After startup, the system enters the Landing state at which it idles until actions are taken by the user. (e.g. Login, Create Listing) After completing a set of actions, the system returns to the idle state.

5. Change & Risk Management Process

The process by which product scope and requirements can be altered is as follows:
1.) During any sprint meeting, any group member may bring up a feature or requirement which they have been assigned and unable to address
2.) The group may then vote to determine whether said requirement will be taken on by another group member, addressed in a different way, placed on the backlog, or removed
3.) If the requirement is determined to be unfeasible and thus set for removal, the SRS manager will update the SRS accordingly
4.) The SRS manager will present the new SRS to the group for approval on the change

A. Appendices