SW Engineering CSC648/848 Summer 2019
Safe SF
Team 2

Team: Local

Alex Wolski
Team Lead & Back End
awolski@mail.sfsu.edu

David Stillwagon (Ryan)
Back End Lead
stillwagonryan@gmail.com

Lidiya Jebessa
Back End
ljebessa@mail.sfsu.edu

Sunny Srijan
Front End Lead
sunnysrijan@gmail.com

Evan Guan
Front End
eguan3@mail.sfsu.edu

Tristan Mclennan
Front End
Tmclennan@mail.sfsu.edu

Harshveer Saini (Harsh)
GitHub Master & Front End/Back End
hsaini@mail.sfsu.edu

Milestone 2
July 12, 2019

Reversion Table:

Version	Date	Description
1.0	July 12	Submitted For Review

1) **Data Definitions V2**

- **Hazard_type (String, Primary Key)**
  List of approved hazard types that a report can be tagged with.

- **Location (String, Primary Key)**
  List of notable locations that a report can be tagged with.

- **Report**
  - **Report_id (Int, Primary Key)**
    The id assigned to the report upon creation
  - **Report_user_id (Int, Foreign Key)**
    Reference to the user in the User table that created the report
  - **Report_details (String)**
    Additional details about the hazard provided by the user
  - **Report_image (String)**
    Filename of image file on server
  - **Report_status (String)**
    Either unassigned, assigned, or completed
  - **Category (Int, Foreign Key)**
    Reference to a type of hazard in the Hazard_types table.
  - **Location (Int, Foreign Key)**
    A reference to a location in the Locations table
  - **Loc_long (Float)**
    longitude of the location being reported
  - **Loc_lat (Float)**
    Latitude of the location being reported
  - **Report_creation_date (DATE)**
    Timestamp of report creation date
  - **Report_update_date (DATE)**
    Timestamp of the last time the report was updated

- **User**
    - **User_id (Int, Primary Key)**
      ID assigned to the user

    - **Status (String)**
      Either registered or admin

    - **Display_name (String)**
      The user's display name

    - **Email (String)**
      The email of the user

    - **Password (String)**
      The hashed password of the user

    - **Report_count (Int)**
      How many reports the user has made

    - **Join_date (DATE)**
      The join date of the user

2. **Functional Requirements V2**

Priority 1

1. Unregistered users shall be able to view existing reports
2. Unregistered user shall be able to filter through existing reports based on category.
3. Registered user shall be able to do everything that an unregistered user can do
4. Registered user shall be able to make new reports for environmental hazards.
5. Admins when logged in with their admin accounts shall be able to change the status of reported hazards after assigning them.

Priority 2

1. Unregistered users shall be able to pan through a map of San Francisco to see the reported hazards marked by pins.
2. Admins shall be able to manage user accounts

3) UI Mockups and Storyboard

Use Case 1: Dylan goes to a beach

Dylan decides to go to China Beach this weekend. He finds a deceased sea lion that has washed up on the beach and wants to report it, but he doesn't want to do it on his phone. So he takes a picture and waits until he gets home to make the report. At home on his PC, he checks for an existing post about the incident by using the hazard browsing feature. Once he confirms that no one has reported it yet, Dylan clicks the report button to create a new report. He is given clearly labeled input boxes to fill out for the location name, hazard category, location on a map, an optional image, and an optional description. On submission, if he is already logged in, the post is created. If not, he is prompted to register or login before he can post.

SF Safe (Home button)

Register/Login/Log off/User Account and Posts

Search

Category Dropdown

Post Report

2) Create a new post

1a) Find the park

Category/Location Filters

1b) Check to see if there is a posting on the problem already. It doesn't, so continue to step 2.

Google Map with recent (in the past week?) report pins

Image

Recent Post Title 1

Location and time (newest) of posting

Short summary

Image

Recent Post Title 2

Location and time of posting

Short summary

Image

Recent Post Title 3

Location and time (oldest) of posting

Short summary

SF Safe (Home button)

Register/Login/Log off/User Account and Posts

Search

Category Dropdown

Post Report

3b) Select Category

Incident Category Selection List

3a) Create Pin

Google Map with single user-settable pin for location setting.

Comments:

3c) Enter a comment or details about the incident

Comments entry

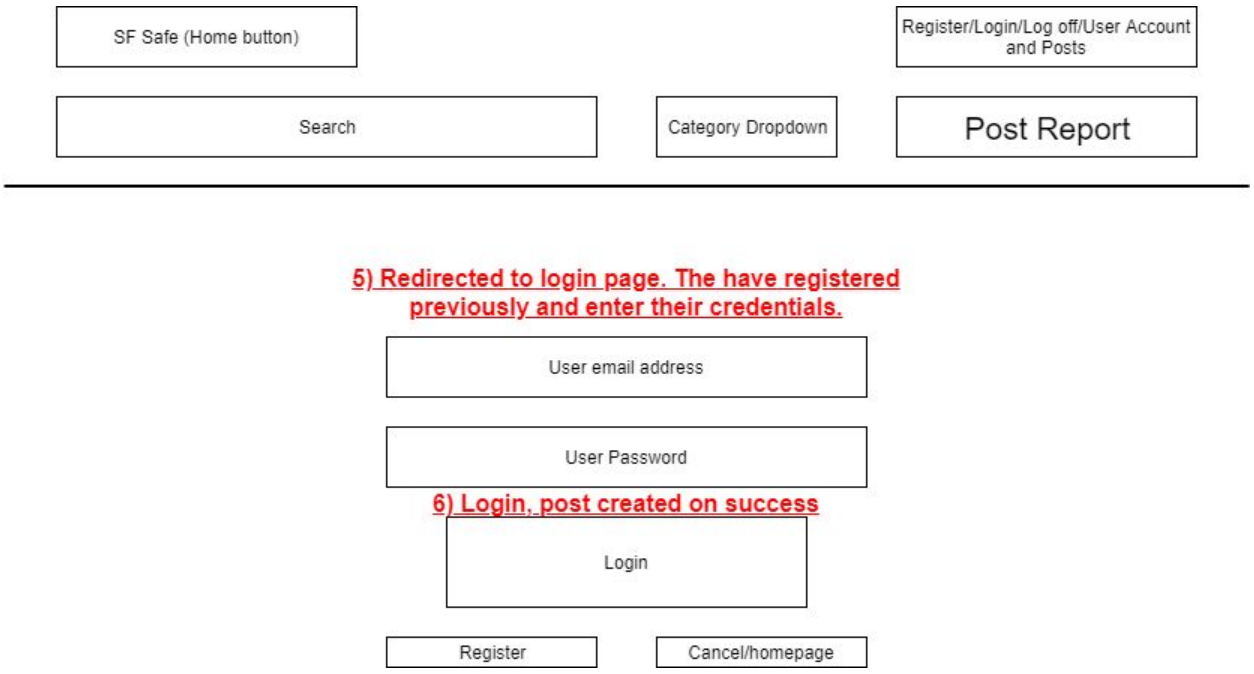
Upload Image:

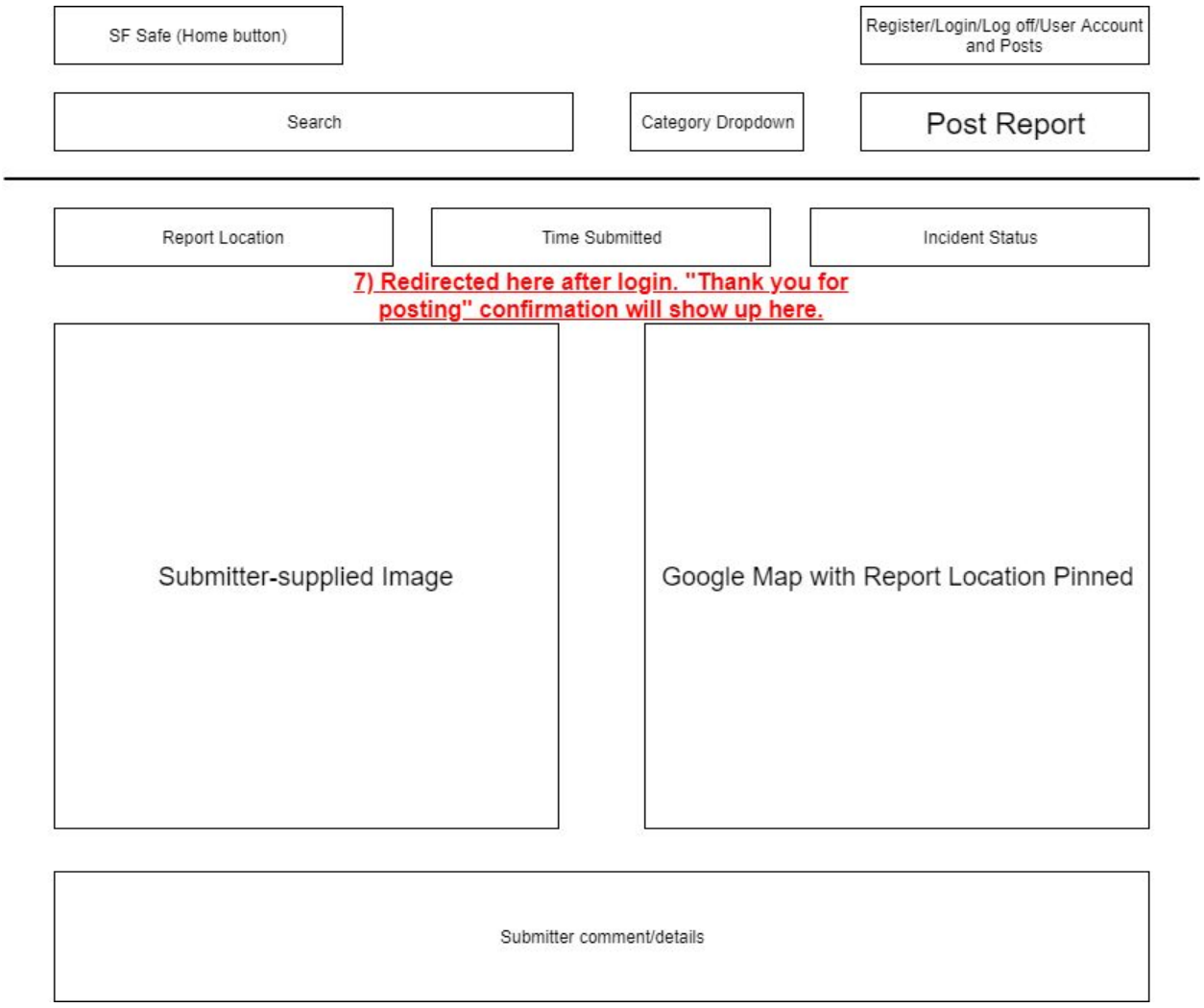
3d) Select and upload an image from the user's computer

Image Upload Widget

4) Finish creating the post and submit it.

Create Report


SF Safe (Home button)
Register/Login/Log off/User Account and Posts
Search
Category Dropdown
Post Report
5) Redirected to login page. The have registered previously and enter their credentials.
User email address
User Password
6) Login, post created on success
Login
Register
Cancel/homepage



Use Case 2: Patricia and Eddie go to The Panhandle

The couple go to the Golden Gate Park Panhandle. They come across a downed tree and would like to report on it. They take a picture of the tree and make a post when they get home. On their PC, they create a new hazard report. They are given clearly labeled input boxes to fill out for the location name, hazard category, location on a map, an optional image, and an optional description. On submission, if they are already logged in, the post is created. If not, they are prompted to register or login.

SF Safe (Home button)

Register/Login/Log off/User Account and Posts

Search

Category Dropdown

Post Report

1) Create a new post

Category/Location Filters

Google Map with recent (in the past week?) report pins

Image

Recent Post Title 1

Location and time (newest) of posting

Short summary

Image

Recent Post Title 2

Location and time of posting

Short summary

Image

Recent Post Title 3

Location and time (oldest) of posting

Short summary

SF Safe (Home button)

Register/Login/Log off/User Account and Posts

Search

Category Dropdown

Post Report

2b) Select Category

Incident Category Selection List

2a) Create Pin

Google Map with single user-settable pin for location setting.

Comments:

2c) Enter a comment or details about the incident

Comments entry

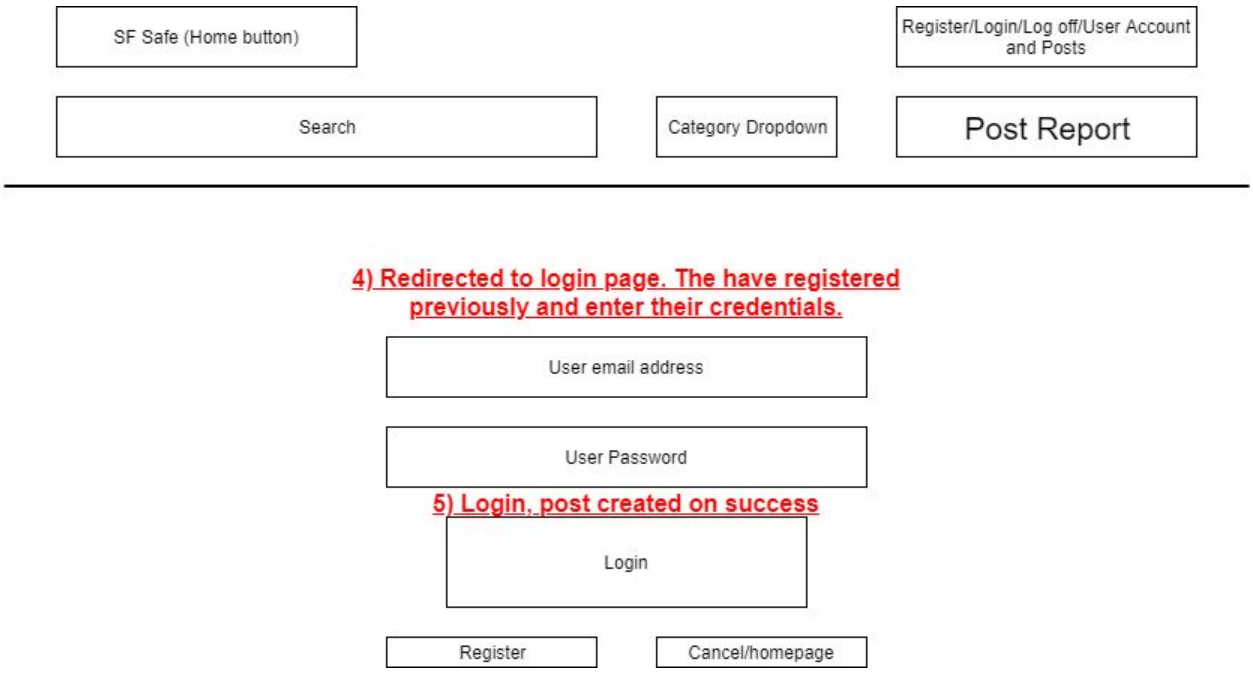
Upload Image:

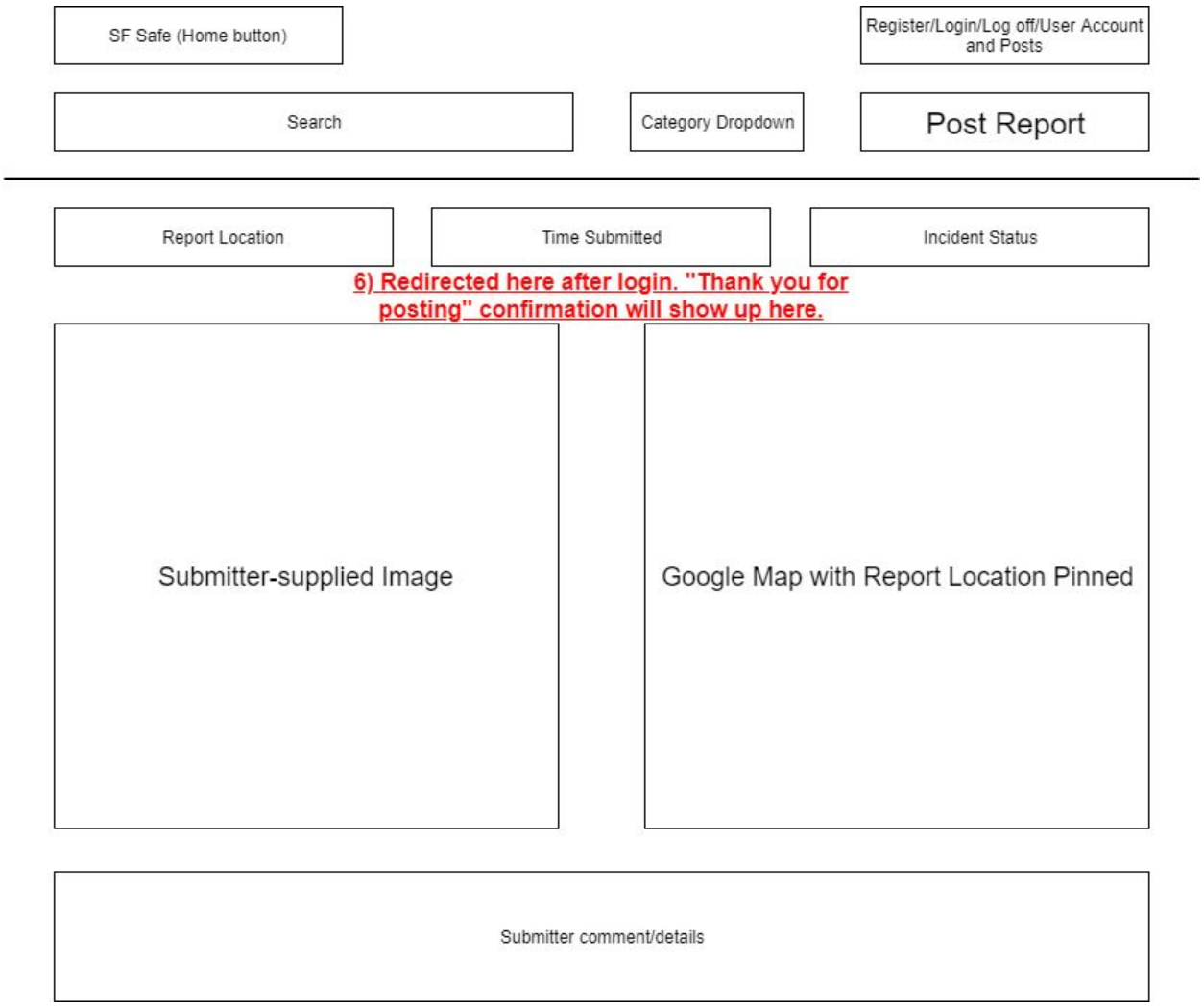
2d) Select and upload an image from the user's computer

Image Upload Widget

3) Finish creating the post and submit it.

Create Report


SF Safe (Home button)
Register/Login/Log off/User Account and Posts
Search
Category Dropdown
Post Report
4) Redirected to login page. The have registered previously and enter their credentials.
User email address
User Password
5) Login, post created on success
Login
Register
Cancel/homepage



Use Case 3: Jenna goes to Fort Funston

Jenna wants to go to Fort Funston today. Since the Fort Funston has record of closing down due to environmental hazards, Jenna wants to check to see if there are any problems before going out. She goes to the hazard browsing page of the site. Jenna then uses the filtering options to restrict the results to Fort Funston. She finds one that there was a mudslide on the trail she was planning to take and alters her course.

SF Safe (Home button)

Register/Login/Log off/User Account and Posts

Search

Category Dropdown

Post Report

2a) Find the park

Category/Location Filters

2b) Check to see if there are any issues at the park she is going to. There is one, so she clicks on the link in the pin's details popup and is redirected its report page.

Google Map with recent (in the past week?) report pins

Image

Recent Post Title 1

Location and time (newest) of posting

Short summary

1) Check to see if there are any recent postings that are relevant to her. Theree aren't any, so she goes to check the map.

Image

Recent Post Title 2

Location and time of posting

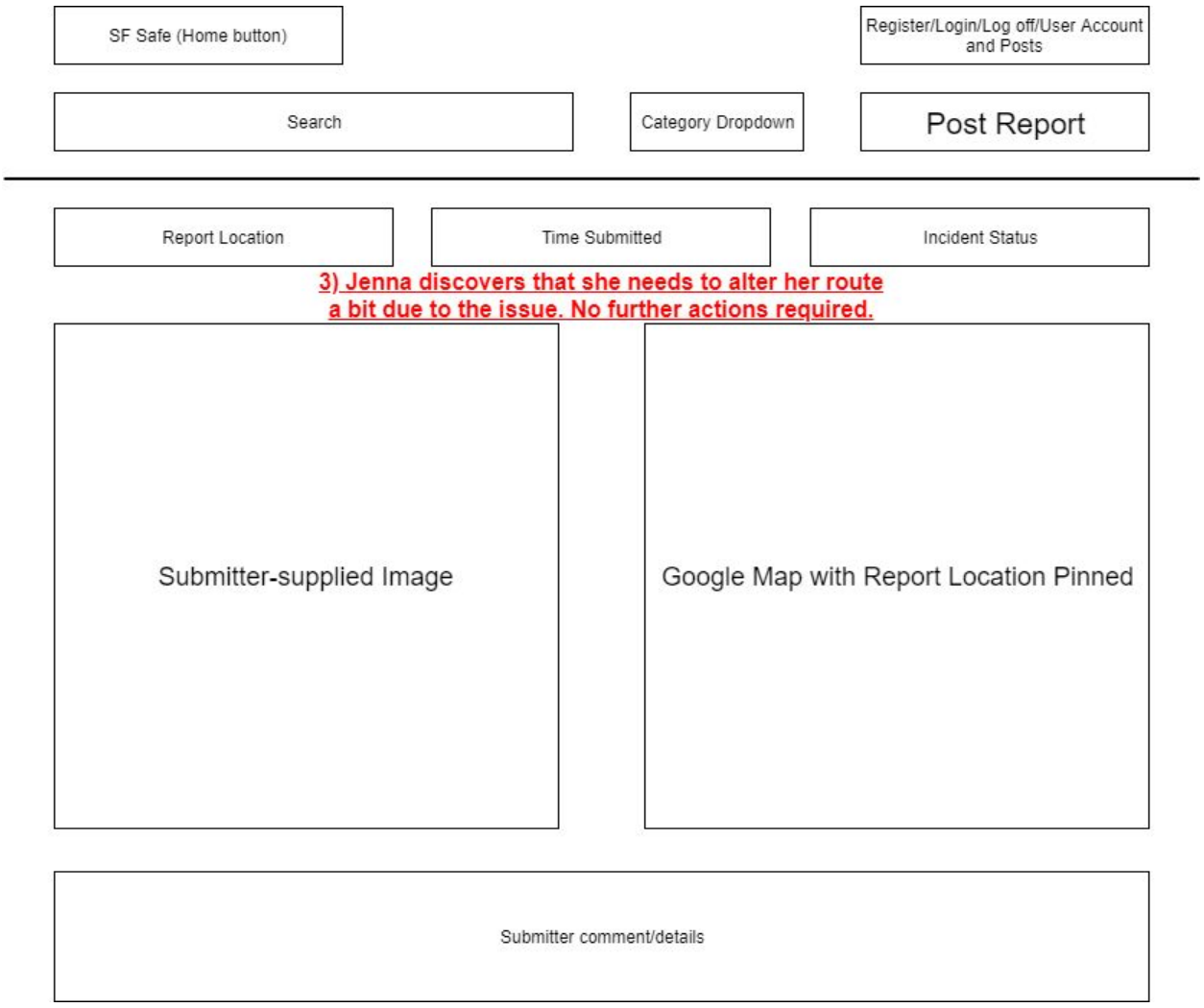
Short summary

Image

Recent Post Title 3

Location and time (oldest) of posting

Short summary



Use Case 4: Kevin's Typical Workday

Kevin's typical workday involves looking at, updating the status of, approving, and rejecting reports made by users of the website. He starts off by logging in to his account using the appropriate function on the website homepage. When he logs in, he is redirected to the homepage, but because he is an admin, he sees different elements than a lower privileged user. His homepage consists entirely of a list of reports that by default is sorted by time and status (oldest, unprocessed reports first), though he can change the sorting and filtering with functions on the webpage. To update the status of a report, he finds a report using filter functions or search, and then uses the appropriate function on the report page to change the status of the report. When he is done for the day, he needs to logoff, and does so with the function on top of all pages.

SF Safe (Home button)

1) Kevin logs in using a login popup widget and the homepage is refreshed.

Register/Login/Log off/User Account and Posts

Search

Category Dropdown

Post Report

Category/Location Filters

Google Map with recent (in the past week?) report pins

Image

Recent Post Title 1

Location and time (newest) of posting

Short summary

Image

Recent Post Title 2

Location and time of posting

Short summary

Image

Recent Post Title 3

Location and time (oldest) of posting

Short summary

SF Safe (Home button)

Register/Login/Log off/User Account and Posts

Search

Category Dropdown

Post Report

Image

Recent Post Title 1

Location and time (newest) of posting

Short summary

Image

Recent Post Title 2

Location and time of posting

Short summary

Image

Recent Post Title 3

Location and time (oldest) of posting

Short summary

Image

Admin Reviewed Post 1

Time (newest) of admin review

Short summary

Image

Admin Reviewed Post 2

Time of admin review

Short summary

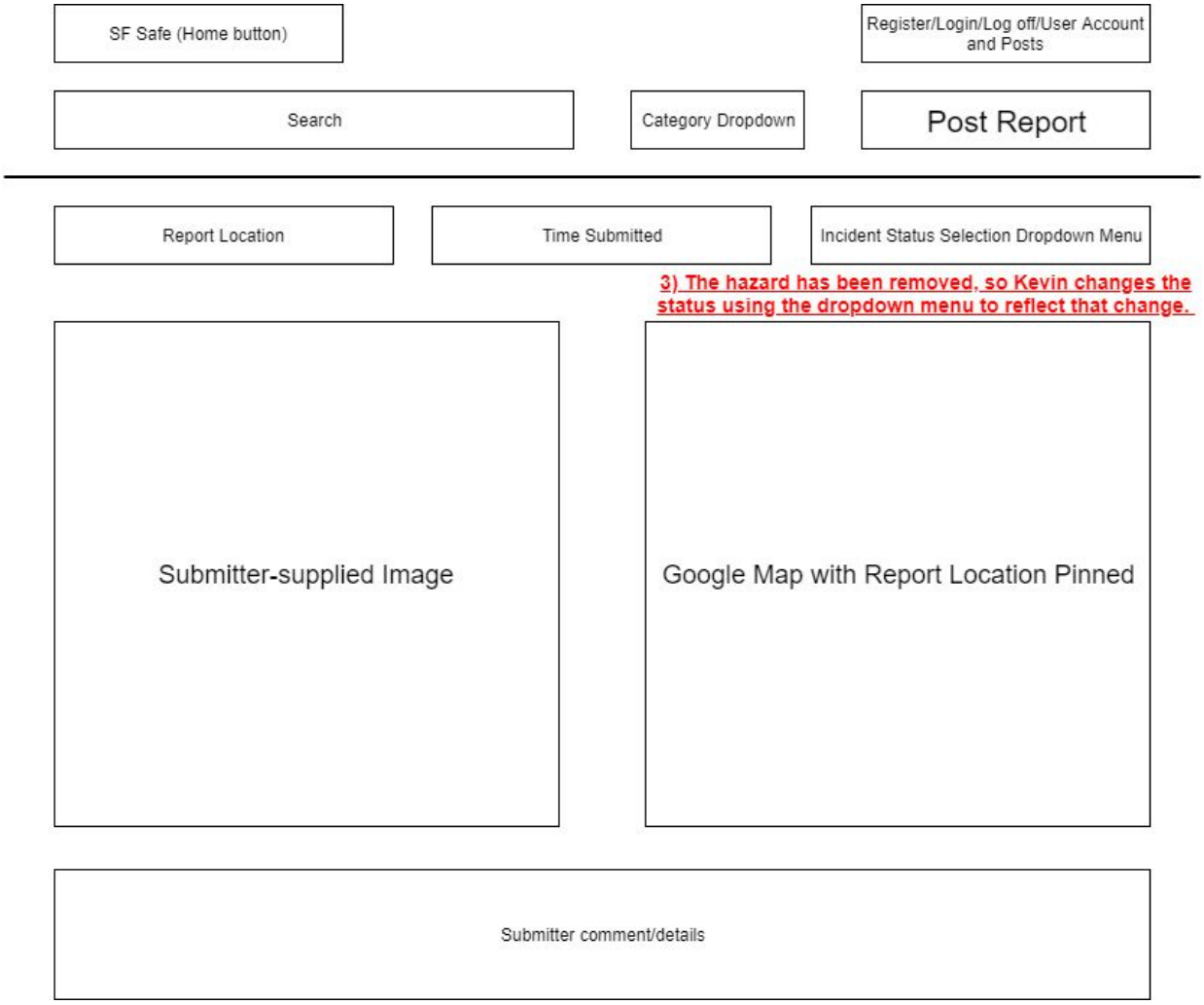
2) Kevin's admin homepage. He needs to follow up on this report, so he clicks on it.

Image

Admin Reviewed Post 3

Time (oldest) of admin review

Short summary


SF Safe (Home button)
Register/Login/Log off/User Account and Posts
Search
Category Dropdown
Post Report
Report Location
Time Submitted
Incident Status Selection Dropdown Menu
3) The hazard has been removed, so Kevin changes the status using the dropdown menu to reflect that change.
Submitter-supplied Image
Google Map with Report Location Pinned
Submitter comment/details

4) High level Architecture, Database Organization

Database Organization:

Our database will contain four tables: **Hazard_type**, **Location**, **Report**, and **User**. The **Hazard_type** and **Location** tables only contain a list of hazards and locations that are referenced in other tables and used to populate drop down menus on the webpage.

The **Report** table contains important information about the hazard and metadata used to search for it. The primary key is the **Report_id**, a unique number generated to represent the report. The **Report_user_id** is an integer corresponding to the user who created the report in the **Users** table. The **Report_details** field is a string containing a description of the hazard provided by the user. The **Report_image** is the filename of the image for the report. The image is stored on the server. The **Report_status** is a string and can only be one of three states: unassigned, assigned, and complete. The **Category** field is an integer corresponding to a one of the categories in the **Category** table. Same with the **Location** field, which points to a location in the **Location** table. **Loc_long** and **Loc_lat** are floating point numbers representing the longitude and latitude of the hazard. And the **Report_creation_date** and **Report_update_date** contain date objects that define the date the report was created and the date is was last updated.

The **User** table contains personal information about the user. The **User_id** is a unique integer identifying the user. The **Status** field is a string which can either be "registered" or "admin." If the user is an admin, they can access the admin tools for moderating the site. The **Display_name** is the username the user chose. The **Email** of the user is stored, but doesn't have a use in this version of the website. The **Password** field contains the hashed password of the user. **Report_count** is a simple counter of how many reports the user has made. And **Join_date** holds the date that the user registered on the site.

Media Storage:

The only media we are storing are images of the hazards. The images are uploaded for the public to see, so we didn't take any measures to secure the images. They are stored in a file on the server as regular image files. Each report is only allowed to have one image attached, so the filenames of the images will be the report id. For the generated thumbnail images, the filename will be the report id followed by "_thumb."

Search and Filter Implementation:

For the filtering, we will request all of the data in one SQL call. The options the user can filter are Location and Category. We can use the SQL call **SELECT * FROM table_name** to select all of the reports. Then we add **WHERE column_name = option** to only include that has

the location or category that we want. So the full SQL call that returns only the reports that matches the users' filtering settings is:

SELECT * FROM Reports WHERE Category = chosen_category AND Location = option

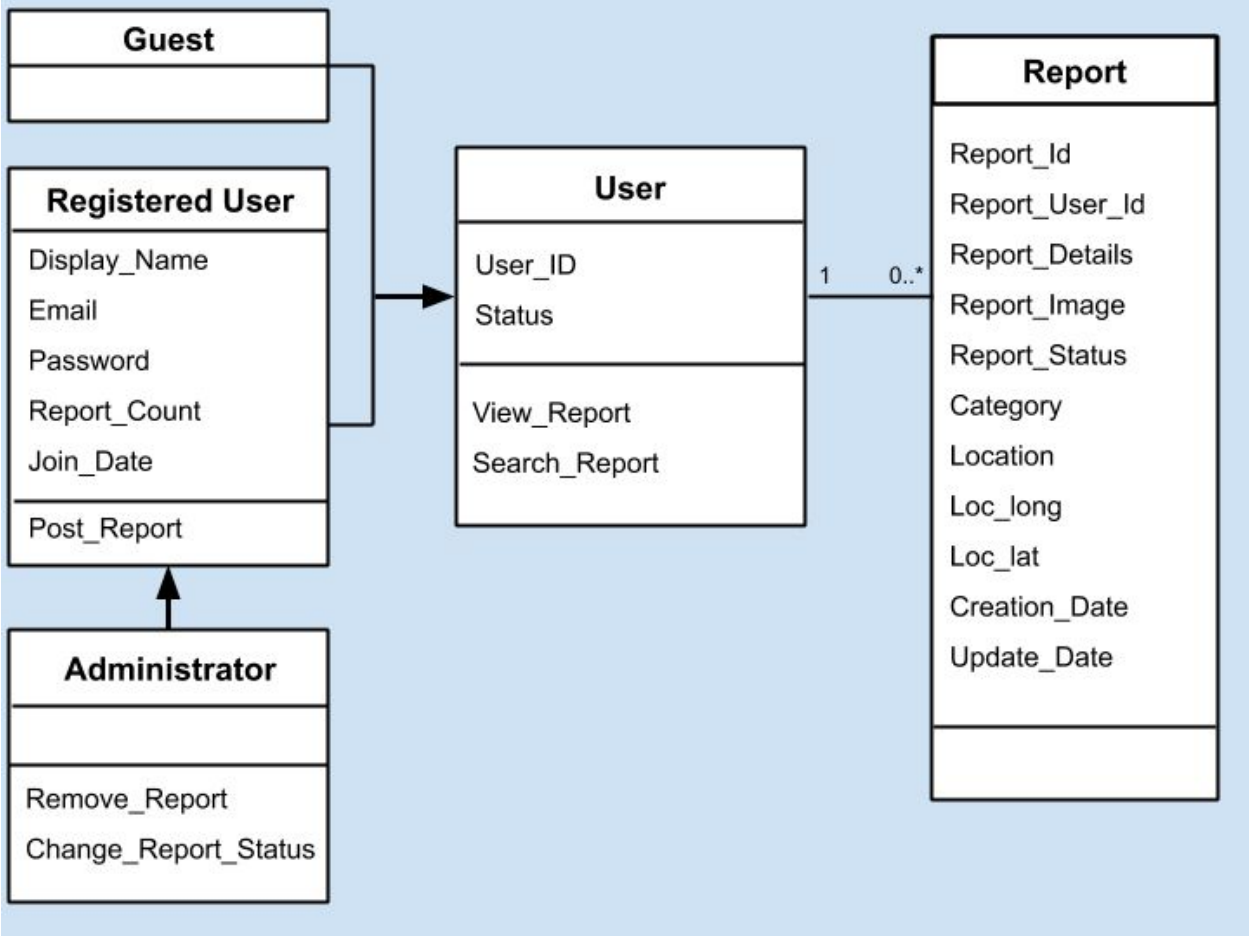
For the text search, we will use the **LIKE** SQL command. This will select the database entries that contain a key phrase. To search a column for a keyword, we would use the command: **SELECT * FROM table_name WHERE column_name LIKE search_term**. The columns we will be searching in are Category, Location, and Report_details. The SQL call that would search these columns is:

SELECT * FROM Reports WHERE Category LIKE "%search_term%" OR Location LIKE "%search_term%" OR Report_detailsLIKE "%search_term%"

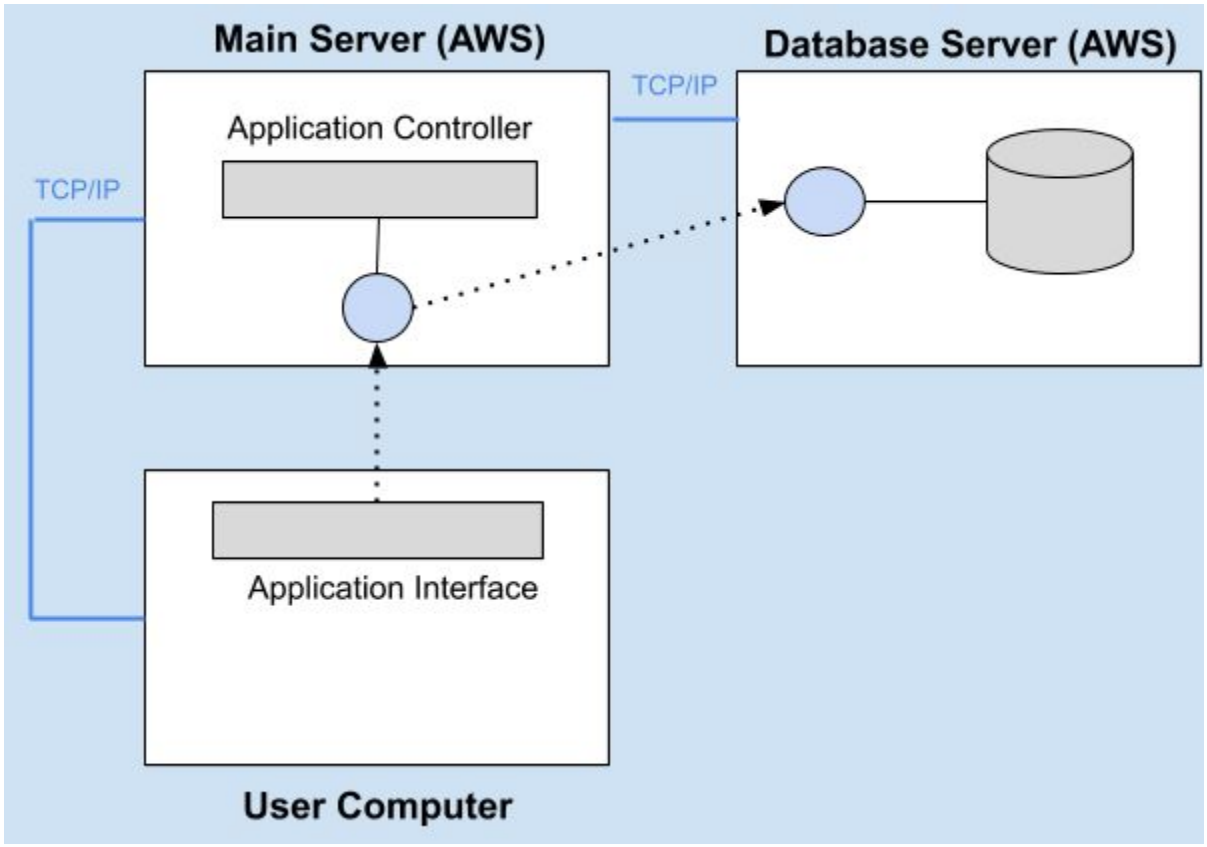
This command can easily be expanded to take multiple search terms, but currently it only takes one.

5) High Level UML Diagrams

Class Diagram:



Component and deployment diagram:



6) Identify actual key risks for your project at this time

1. Understand the existing system that is in place by the city
2. Even though this is not a live project, ensuring that the data we store and our system is secure.
3. Lack of experienced front end developers and the pressing time limits might demand some focussed code sprints.

7) Project management

For the management of our team, we have been using Trello. We assign tasks to each member based on what else they are working on and their strongpoints. We have been careful to assign everyone a task that doesn't depend on another person's work so that we are all working concurrently. Unfortunately, none of us are experienced enough to accurately guess the duration of each task. So we have been assigning tasks only in the short term. And once a member completes a task, they are assigned a new one. This can be a problem since members don't have a deadline and may become complacent. In addition, the team doesn't have a good idea of how close to completion we are. So we can't alter our schedule or request an extension until closer to the deadline.

In the future, we plan to break down our objective into all of the tasks needed to complete it instead of creating tasks as we go. In addition, we will guess how long each task will take and plan out our time usage until the deadline.