M2 Deliverable

LifeCycle Thugs

February 16th, 2024

What is our project?

Our group will be working on the IOT project. Our project “QuakeQuest” is a web application that displays visual earthquake information. It will use a US government API for data.

A high-level description of what our project will offer.

QuakeQuest is a web app that will deliver data visualization, data prediction, data filtering and map plotting of past earthquakes using the United States Geological Survey (USGS) Earthquake Hazards Program API. The website will include user features such as a registration system, notification system and dashboard. Its primary purpose is to enhance public safety by offering timely information about seismic events, allowing individuals, communities, and organizations to prepare and respond effectively to potential threats.

User Requirements:

1. Users will be able to view earthquakes displayed on a map.
2. Users will be able to filter the data by selecting options such as date, location, size, etc.
3. Users will be able to share their results with others (social media sharing options).
4. Users will be able to toggle different map views (such as satellite, terrain, etc).
5. Users will be able to log into the system with a username and password.
6. Users will be able to view and edit their profile.
7. Users will be able to receive custom notifications from the system (such as a new earthquake of certain conditions), as well as removing notifications via phone/email.
8. Users will be able to view earthquake predictions.

Functional Requirements:

1. The application will get data from the USGS earthquake API, and then store the data within a database so that we can minimize the amount of requests from the API and provide quicker results to our users.
2. The application will use the data stored in the database to display the data on a global map interactive map.
3. The application will provide input fields so that users enter parameters that filter the data. This will be done converting user input into a database query.
4. The database will store a user’s information such as login, password, and other fields.

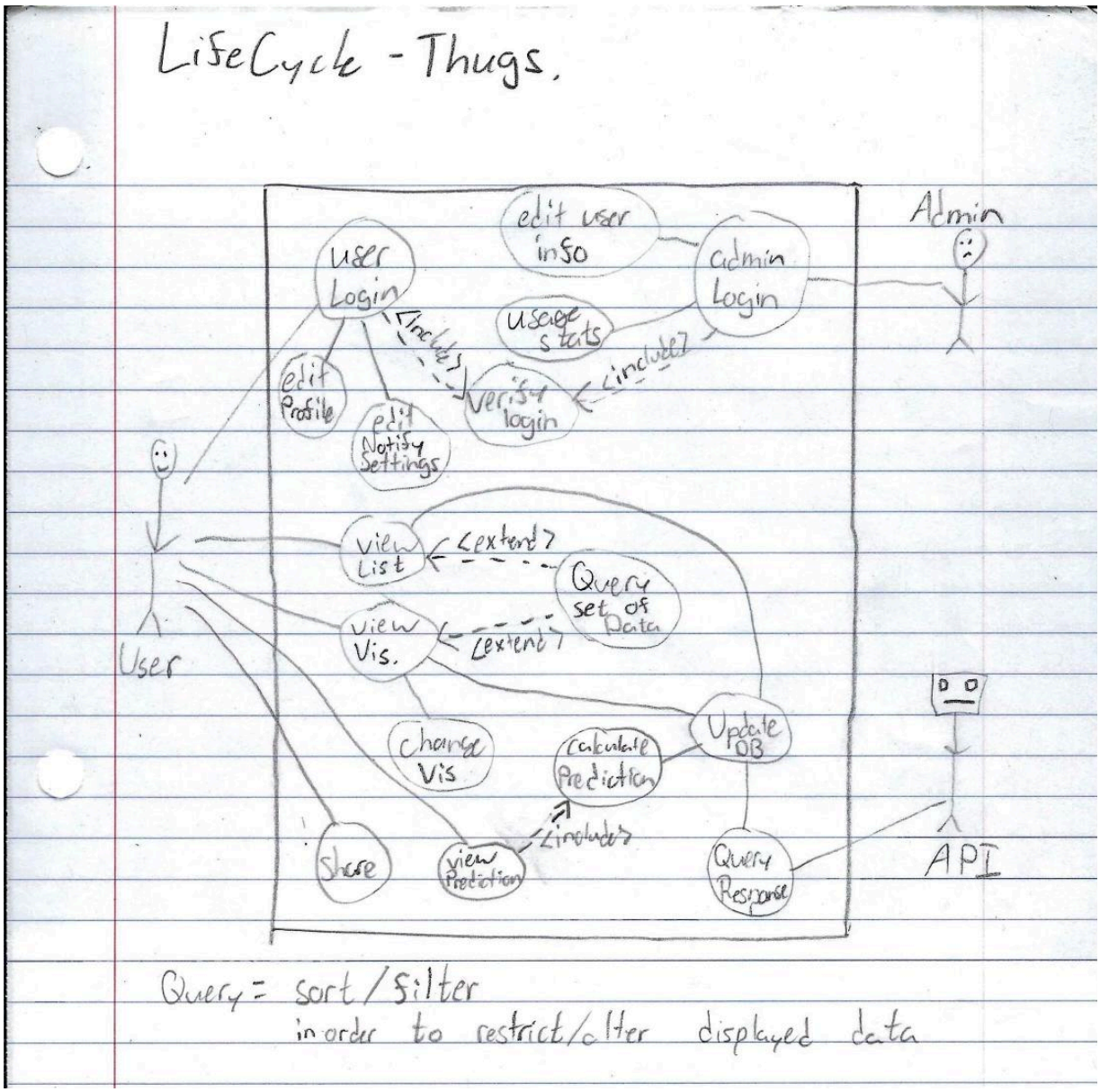
5. The application will provide social media links that will share queried search results (ex: if a user searches for all earthquakes between Feb 2 and Feb 17 2023, they will be able to share the link so that another user can view the same page and map).
6. The application will provide data prediction for future events.
7. The system will automatically send custom notifications to the users if the conditions (set by the user) have been met.
8. An admin will be able to view and delete users and their notifications (for moderation purposes).
9. The system will automatically send out password reset emails for users that have forgotten their password/login.
10. The system will display the number of API calls and last time the API was called in an administration page.

Non-functional Requirements:

1. The backend programming will be done in python (flask).
2. The system should be able to respond to input and refresh in a few seconds.
3. The system will not display any unnecessary data to other users or administration.

Use Case Diagram & Descriptions

Use Case Diagram



Use Case Descriptions

1. **View data**
   Primary actor: User
   Description: Describes the process when a user lands on website
   Precondition: User connects to site
   Postcondition: User views visualization
   Main scenario:

1. User accesses site via link or search
2. Webpage loads visualization
3. Webpage queries data from database and populates visualization

Extensions:

2a. Improper load, issues error message requesting page refresh.
3a. Fails to properly access database, requests refresh.

2. Filter data

Primary actor: User
Description: Describes the process to filter the data displayed
Precondition: User successfully connects to site
Postcondition: User views altered visualization
Main scenario:

1. User selects an attribute to filter by
2. User selects a condition, (<,>,interval)
3. User sends request
4. Site applies filter to data matching user entry
5. Site updates visualization with new dataset

Extensions:

3a. Request faulty, due to poor format or other, please try again.
5a. Fails to properly load visualization, requests refresh.

3. Share Results

Primary actor: User
Description: Describes the process of sharing results
Precondition: User successfully views and customizes visualization
Postcondition: User posts link to the site with all settings applied
Main scenario:

1. User customizes visualization
2. User clicks link to one of displayed social sites
3. Webpage copies a link to the user's clipboard containing all current page settings.
4. Webpage redirects the user to the selected social network in a new tab.

Extensions:

3a. Faulty link is generated, user unable to share.

4. Login

Primary actor: User
Description: Describes the process to login
Precondition: User accesses website
Postcondition: User is logged in
Main scenario:

1. User selects login
2. User is prompted for login info
3. User enters credentials

4. Entry is checked against the database
5. User is logged in

Extensions:

2a. user may select register account
2a1. user enters new info
2a2. new user added to database
4a. records do not match, error message to please try again.
4b. User is an admin account.
4b1. Update landing page with admin functionality.

5. **Layer Map**

Primary actor: User
Description: Describes the process layering map
Precondition: User successfully views visualization
Postcondition: Different map layers are displayed
Main scenario:

1. User selects layers menu on visualization
2. User toggles one or more settings options
3. Webpage updates its display without refetching data

Extensions:

2a. Improper load, issues error message requesting page refresh.

6. **View User info**

Primary actor: Admin
Description: Describes the process of an admin viewing user info.
Precondition: Admin login successful
Postcondition: Admin accesses user info
Main scenario:

1. Admin select view users option
2. Page fetches user list from database
3. Admin selects a user's account
4. Page fetches specific user info from database
5. Admin views information

Extensions:

2a,4a. Fetch fails, refresh requested.
5a. Admin may enter new values for fields
5a1. Database is updated with new data.
5a2. invalid entry for field, request new entry.

7. **Access usage stats**

Primary actor: Admin
Description: Describes the process of an admin accessing usage stats.
Precondition: Admin login successful
Postcondition: Admin accesses usage stats

Main scenario:

1. Admin select view usage option
2. Page fetches stats from database
3. Page displays usage stats over time.

Extensions:

2a. Fetch fails, refresh requested.

8. **Add Notifications**

Primary actor: User
Description: Describes the process of a User adding a notification.
Precondition: User login successful
Postcondition: User registers a new notification
Main scenario:

1. User selects add notification
2. Page presents user with filters for which events should trigger notifications
3. User selects options and submits
4. Page sends data to the database for storage
5. System checks on update if new information matches user filters

Extensions:

4a. Improper entry, request re-entry from user.
5a. If matched, notification sent.

9. **Send Notifications**

Primary actor: system
Description: Describes the process of the system sending notifications to the users
Precondition: User has registered notifications
Postcondition: User receives notification
Main scenario:

1. The system will receive data from the API
2. System will run an analysis on the newly fetched data
3. System will check if any of the custom notifications have fulfilled conditions
4. System sends notifications to the appropriate user

Extensions:

2a. no response, try again soon
3a. No match, no notifications sent

10. **View Account Settings**

Primary actor: User
Description: Describes the process of a User viewing account settings.
Precondition: User login successful
Postcondition: User accesses their account settings
Main scenario:

1. User selects account settings

2. page queries database for records matching this user.
3. Page presents user with all currently active notification settings and recorded user info
4. User accesses all account settings.
5. If edited, refresh the display.

Extensions:

4a. The user may select edit, Is prompted for input, database updated.
4a1. improper entry, try again.

11. View Prediction

Primary actor: User

Description: Describes the process of a User viewing a prediction.

Precondition: User login successful

Postcondition: User views prediction

Main scenario:

1. User selects view prediction
2. page queries database for data
3. Calculation is performed to make a prediction.
4. Page is updated with prediction

Extensions:

2a. Query fails, please refresh
3. Insufficient data, try again later.