Software Requirements and Design Document

For

Group 10

Version 1.0

Authors:
Samuel Anderson
Alex Gonzalez
Claudio Olmeda Florio
Juancarlos Alguera
Joel Bustamante

1. Overview

Mapapp is a social media platform that is intended to allow users to view and create content without traditional, algorithmic feeds, but instead geographically. That is, posts appear as circles where they were posted and popularity isn't just measured in likes, but in the actual physical size of the circle. To make the site still useful, traditional features from other social media platforms are present, such as commenting, liking, viewing old posts.

2. Functional Requirements

User Requirements

1. The user shall be able to create an account
2. A signed in user shall be able to create posts
3. A signed in user shall be able to comment on posts
4. A signed in user shall be able to like posts
5. The user shall be able to change his username, email, and password
6. The user shall be able to sign out at any time

Post Requirements

1. Posts should be displayed in their geographic location as circles
2. Posts consist of text content and a color
3. The more likes a post has, the larger its radius

Map Requirements

1. The geolocation of the mouse shall be displayed
2. There shall be a home button that redirects the user to their location
3. The site shall ask for permission to the user's location

3. Non-functional Requirements

Performance Requirements

1. The map shouldn't take more than 5 seconds to load
2. Signing in should refresh all components

Compatibility Requirements

1. The app should work the same on Firefox, Chrome, and Safari
2. The app should work on mobile

Usability Requirements

1. On smaller screens, the UI should be adapted for easier usability
2. Creating an account should immediately sign in the user
3. The comments sheet shall not take up too much space on the screen to allow for concurrent navigation

Post Requirements

1. The formula for the size of a post's visual circle is in proportion to the number of likes it has over the highest number of likes on a post multiplied by some constant
2. To open a post, the user clicks on a post bubble and a popup appears
3. To see the comments on a post, the user clicks the comments button and a sidebar appears with a list of comments and the post's content w/ the color
4. The post button shall be displayed on the bottom left corner of the screen

5. Clicking the post button opens a popup where the user can type text and select a color from a slider

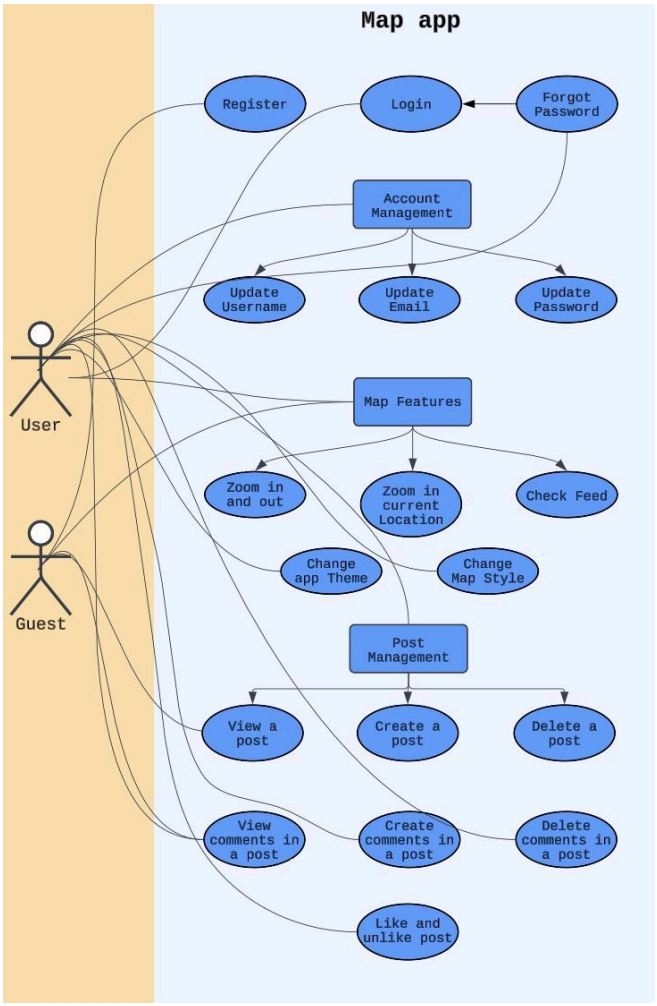
User Requirements

1. The map shouldn't take more than 5 seconds to load
2. To sign out, the user clicks the account button on the top bar and clicks sign out

Security Requirements

1. Signing in should be authenticated with a token and CSRF cookie
2. Cookies should have SameSite="Strict" to avoid CSRF

4. Use Case Diagram (10 points)



Textual descriptions of use cases:

Use Case: Register
Actors: Guest
Preconditions: Guest has accessed the registration page. No duplicate email or username exists in the system.

Main Flow: The guest will provide a username, an email and a password. The system will validate the input, create a new account and store the details. User gets redirected to the login page.
Alternative Flows: If validation fails like an invalid email an error message is displayed, and the guest is asked to retry.
Postconditions: A new user account is created.

Use Case: Login
Actors: User
Preconditions: User is on the login page and the account exists.
Main Flow: User gives the correct email and password. The system then authenticates the credentials. User is logged in and redirected to the main map feed.
Alternative Flows: If credentials are invalid, an error message is displayed and the user is asked to try again.
Postconditions: User

Use Case: Forgot Password
Actors: User
Preconditions: User is in the "Forgot Password" page. The email entered exists in the database.
Main Flow: User enters their email. The system sends a password reset link to the email. User follows the link and resets their password.
Alternative Flows: If the email is not registered, an error message is displayed.
Postconditions: User resets their password and can log in with the new one.

Use Case: Update Username
Actors: User
Preconditions: User is logged in, and in a valid session.
Main Flow: User navigates to the account settings page by going into Account -> Update User. User enters their password and clicks on the Update Username button. They enter a new username. The system checks if the username is in the database. Success message pops up.
Alternative Flows: If validation fails like if the username exists an error message is displayed, and the user is asked to choose a new name.
Postconditions: User's username is updated in the system

Use Case: Update Email
Actors: User
Preconditions: User is logged in, and in a valid session.
Main Flow: User navigates to the account settings page by going into Account -> Update User. User enters their password and clicks on the update email button. System checks email format then updates the email in the database.
Alternative Flows: If validation fails like email already exists an error message is displayed.
Postconditions: User's email is updated.

Use Case: Update Password
Actors: User
Preconditions: User is logged in, and in a valid session.
Main Flow: User navigates to the account settings page by going into Account -> Update User. User enters their password and clicks on the update password button. System validates the new password then updates the password in the database.
Alternative Flows: If the current password is incorrect or the new password does not meet requirements, an error message is displayed.
Postconditions: User's password is updated.

Use Case: Zoom in and out of the map
Actors: Guest and Users
Preconditions: User or Guest has accessed the home page.

Main Flow: The actor interacts with the map zoom controls, by either clicking on the + and - icon, or by using scroll. The system adjusts the zoom level of the map according to the input.
Alternative Flows: If the map is already at the maximum or minimum zoom the zooming actions are disabled.
Postconditions: The map's zoom level is updated

Use Case: Go to current location
Actors: User and Guests
Preconditions: Actors allowed location permissions to the application
Main Flow: The actor clicks on the home icon on the top left in the navbar. The system retrieves the user's approximate location. The map zooms in near the actor's current location.
Alternative Flows: If location permissions are denied the system displays an error message prompting the user to enable location access. If location services fail an error message is displayed.
Postconditions: The system zooms in near the actors location and the map is updated. Otherwise an appropriate error message is shown.

Use Case: Change apps Theme
Actors: User
Preconditions: User is logged in, and in a valid session.
Main Flow: User clicks on the settings icon on the top right. The system displays a pop up menu with the settings. User flips the theme switch on or off indicating light or dark theme. The system applies the selected theme to the user interface.
Alternative Flows: If the theme selection fails to apply an error message is displayed.
Postconditions: The user interface reflects the newly selected theme.

Use Case: Change Map Style
Actors: User
Preconditions: User is logged in, and in a valid session.
Main Flow: User clicks on the settings icon on the top right. The system displays a pop up menu with the settings. Next the user selects the desired style from the dropdown list. Once selected, the system updates the map to reflect the new style.
Alternative Flows: If the map style fails to load an error message is displayed and goes back to default.
Postconditions: The map is updated with the new style.

Use Case: View a Post
Actors: Guest and User
Preconditions: The map interface is loaded, and posts have been made.
Main Flow: The actor can see user posts by clicking on any post circle they see on the map. Once a circle is selected, the system retrieves the content of the post. The post's content is displayed in a pop-up.
Alternative Flows: If the post fails to load an error message is displayed.
Postconditions: The actor views the content of the selected post.

Use Case: Create a Post
Actors: User
Preconditions: User is logged in and accessed the map interface, and allowed their location to be shared. .
Main Flow: The user clicks on the notebook icon on the bottom right. The system displays a form for the user to enter the posts content. The user provides the post content and chooses the color of their circle and submits the form. The system validates the input and associates the post with the user's approximate location. The post is saved in the database, and the map is updated with the new circle for the post.

Alternative Flows: If input validation fails an error message is displayed, and the user is asked to try again. If the user hasn't accepted the location service an error message will display. If a guest tries to create a post a message telling them they aren't allowed to is displayed.
Postconditions: The post is created and the map is updated to display the new post.

Use Case: Delete a Post
Actors: User
Preconditions: User is logged in, the post exists, and the user is the original creator of the post.
Main Flow: User navigates to “my posts” under Account in the nav bar. The user clicks on the trash can icon. The system sends the user a confirmation message if accepted the post gets deleted from the database and the circle gets removed. If not, the post remains untouched.
Alternative Flows: If the user does not have permission an error message is displayed indicating they cant delete it. If the post cannot be deleted due to a system error an error message is displayed.
Postconditions: The selected post is deleted, and the map is updated.

Use Case: View Comments in a Post
Actors: Guest and User
Preconditions: The map interface is loaded, and comments have been made in a post.
Main Flow: The actor selects a post on the map. The system gets the posts content and displays it on a pop up. The actor selects the comment icon. The system gets the comments for the post and displays it on a side panel to the right.
Alternative Flows: If comments fail to load an error message is displayed.
Postconditions: The actor views the comments associated with the selected post.

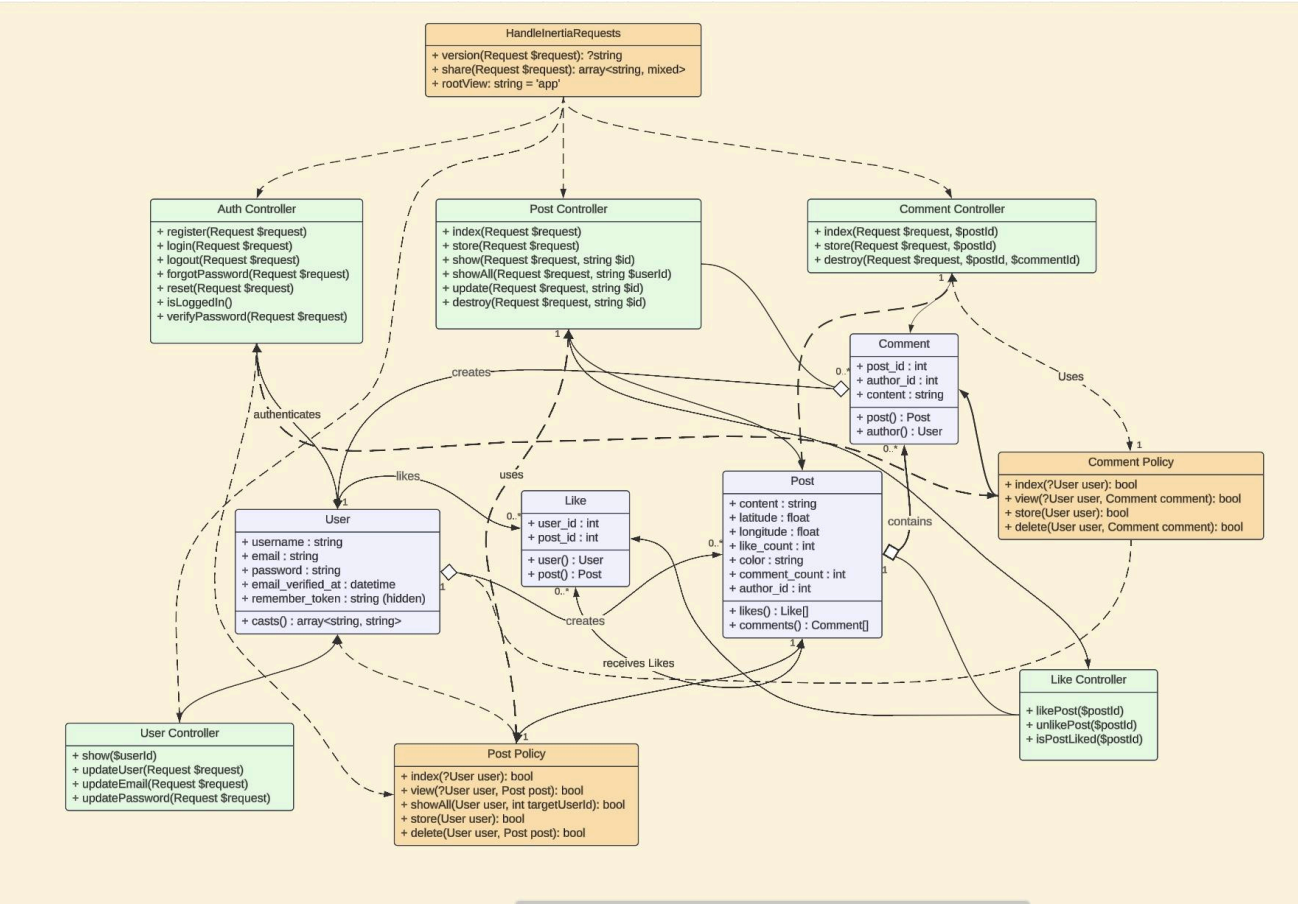
Use Case: Add a Comment to a Post
Actors: User
Preconditions: User is logged in, the map interface is loaded, and the post has been selected.
Main Flow: The user selects a post on the map. The system displays the post content in a pop-up. The user selects the comment icon. The system opens a side panel displaying existing comments. The user types up their comment and clicks on the Submit Comment button. The system validates the input and associates the comment with the selected post. The comment is saved in the database and displayed.
Alternative Flows: If input validation fails an error message is displayed. If the actor does not have permission an error message is displayed.
Postconditions: The comment is successfully added to the post and displayed in the comments section of the side panel.

Use Case: Like a Post
Actors: User
Preconditions: The map interface is loaded, and the post exists.
Main Flow: The actor selects a post on the map. The system displays the post content in a pop-up. The actor clicks the like button in the pop-up. The system records the like in the database and makes the circle bigger.
Alternative Flows: If the like cannot be saved an error message is displayed.
Postconditions: The post's like count is updated, the circle gets bigger, and the system registers the user interaction.

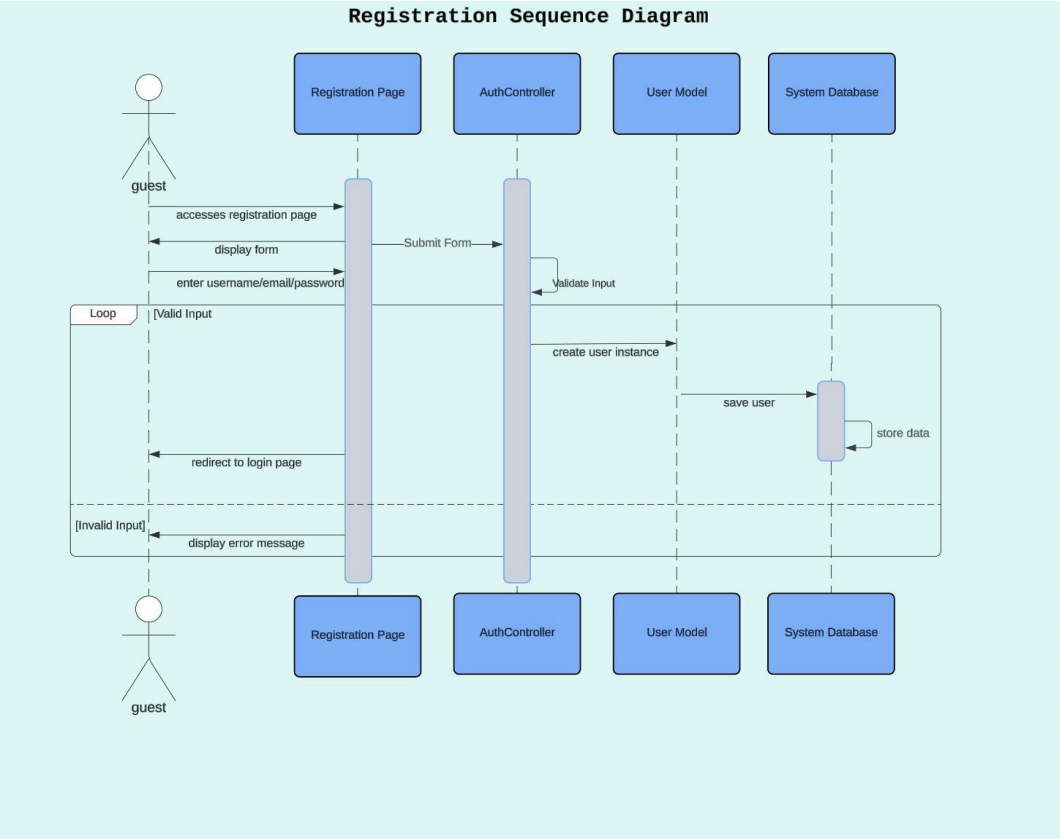
Use Case: Unlike a Post
Actors: User
Preconditions: The map interface is loaded, and the post exists. And the user has already liked the post.
Main Flow: The actor selects the post they liked on the map. The system displays the post content in a pop-up. The actor clicks the like button that is red in the pop-up. The system records the unlike in the database and makes the circle smaller.
Alternative Flows: If the unlike cannot be saved an error message is displayed.

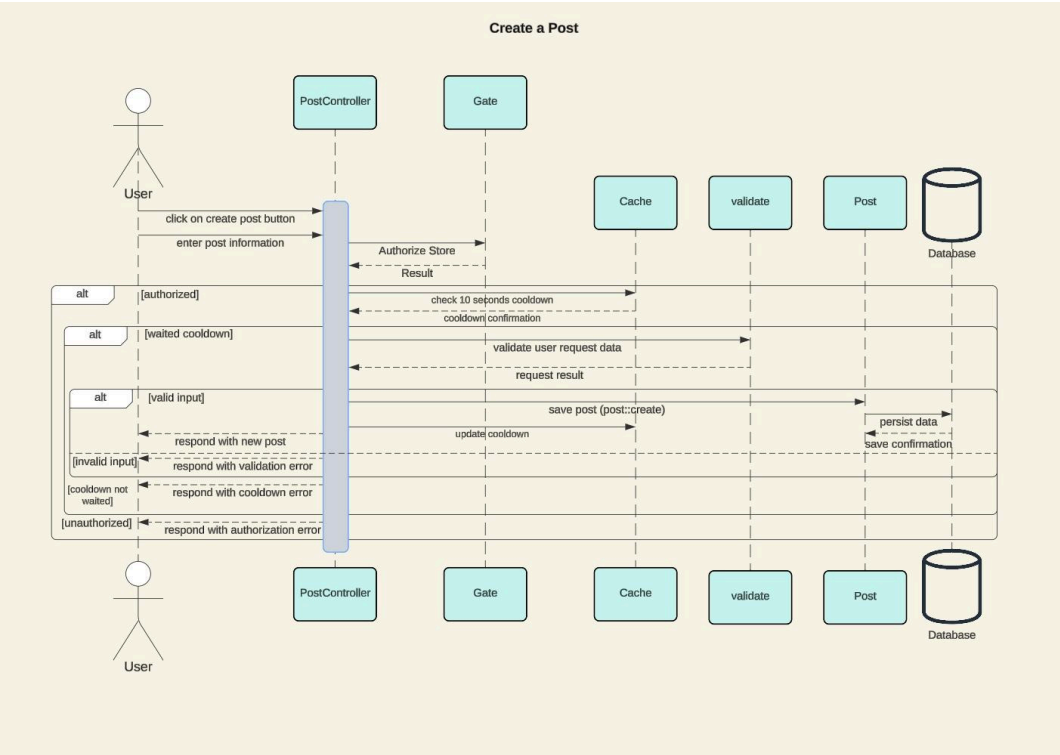
Postconditions: The post's like count is updated, the circle gets smaller, and the system registers the user interaction.

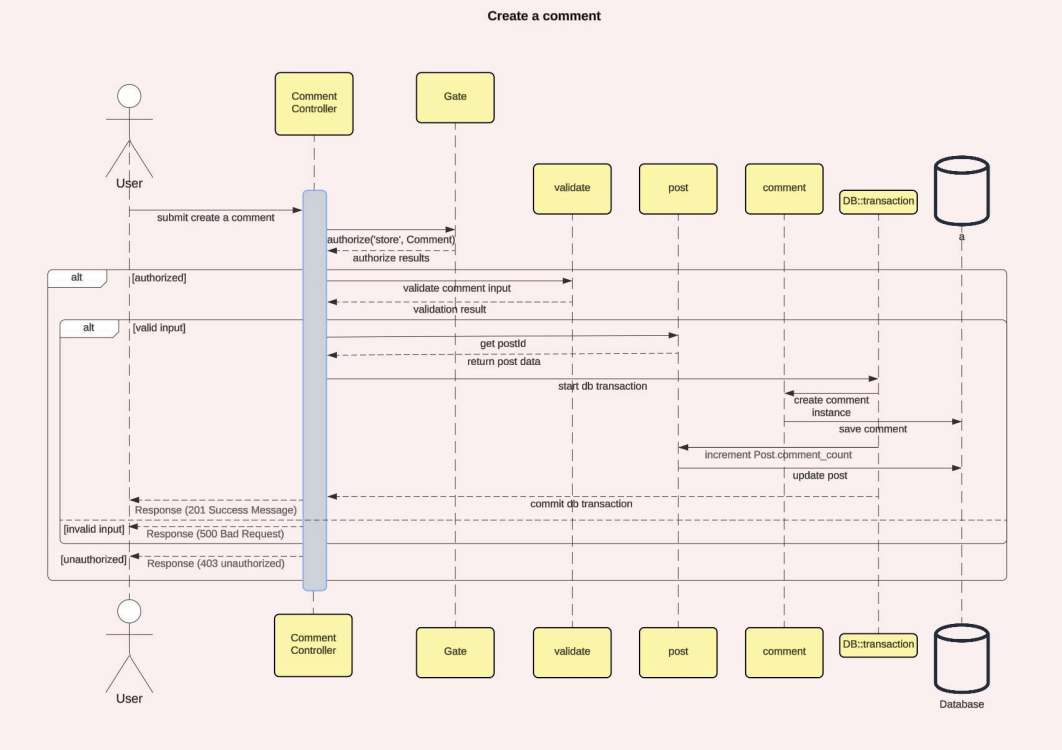
5. Class Diagram and/or Sequence Diagrams (15 points)



Sequence Diagram:







6. Operating Environment (5 points)

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

The program consists of two major components: the **react frontend** and the **laravel backend**. These communicate through **InertiaJS**. Since the frontend uses Javascript and the backend uses PHP, both **npm** and **composer** are required.

The program was successfully run on Microsoft Windows 11 and MacOS Ventura 13.2.1. It could easily be run within a Docker container running Alpine Linux and be fully distributed.

7. Assumptions and Dependencies (5 points)

Assumptions

This project assumes map data exists for every user provided by Leaflet. It also assumes the user will enable location sharing.

Major dependencies (any version marked with '+' implies later versions will work too):

1. Node v22
2. npm (any version that works with Node v22)
3. composer (any version that works with InertiaJS v1.2.0+)
4. IntertiaJS v1.2.0+
5. React v18.3.1