3311-001 (0930-1050) TEAM 10

Our goal is to enrich the UTA student experience with an application that promotes student interaction that utilizes existing geolocation and a form of user verification. With an initial market of on campus locations, the scalability goal will be to allow for adjacent areas and suggested locations based on popularity, preferences, and viability. Users will have a profile that may be shared that displays preferences.

User feedback may be achieved via in class discussion, discord groups with on campus students, or via events through networking. Family and friends are not necessarily the target group; however their insight could prove useful.

To create a web application where users can mark locations on a map, and have profiles with capabilities for linking and friending other profiles, you need a comprehensive approach that includes frontend development, backend development, and user management. Here's a detailed breakdown of what’s required:

**1. Frontend Technologies**

* **HTML/CSS:** For the basic structure and styling of your application.
* **JavaScript:** For interactive elements. Libraries or frameworks like React, Vue.js, or Angular can facilitate dynamic interactions and state management.
* **Map Integration:** Use a mapping library like Leaflet or Google Maps JavaScript API to display and interact with map images or live maps.
* **Canvas/SVG:** For custom overlays and pin placement if not using interactive maps. HTML5 <canvas> or SVG can be used to draw pins or other markers.

**2. Backend Technologies**

* **Server-side Language:** To handle business logic, user management, and data storage. Options include Node.js (with Express), Python (with Flask or Django), Ruby (with Rails), or Java (with Spring Boot).
* **Database:** To store user profiles, friend relationships, pin locations, and other relevant data. Options include SQL databases like PostgreSQL or MySQL, or NoSQL databases like MongoDB.

**3. User Authentication and Profiles**

* **Authentication System:** Implement user authentication to manage user profiles. This can be done using OAuth, JWT (JSON Web Tokens), or other authentication strategies.
* **Profile Management:** Users should be able to create, view, edit, and manage their profiles, including profile pictures, bios, and other personal information.
* **Friendship System:** Implement a system for users to link or friend other users. This involves managing friend requests, acceptances, and possibly mutual friends.

**4. Map Integration**

* **Map Display:** Use a map library to display maps and allow users to interact with them. Libraries such as Leaflet or the Google Maps API provide various features and customization options.
* **Pin Placement:** Allow users to place pins on the map. This involves capturing map coordinates and rendering pins at those locations.

**5. Pin and Location Management**

* **Pin Placement and Management:** Capture user actions (like clicks) on the map to place pins, and provide options to view, edit, or delete pins.
* **Data Storage:** Store pin data in the database, including coordinates, user ID (to associate pins with users), and any additional information.

**6. Friendship Management**

* **Friend Requests:** Implement functionality for sending, receiving, and managing friend requests.
* **Mutual Friends:** Display mutual friends or connections if applicable.
* **Friendship Data Storage:** Store friend relationships in the database, including pending and accepted requests.

**7. User Interface Design**

* **Map UI:** Display the map and allow users to place and interact with pins.
* **Profile UI:** Provide interfaces for viewing and editing user profiles, managing friends, and accessing user settings.
* **Friend Management UI:** Display friend requests, friends list, and allow users to search for and add friends.

**8. Backend API**

* **Authentication Endpoints:** For user login, registration, and profile management.
* **Profile Endpoints:** For creating, updating, and retrieving user profiles.
* **Pin Management Endpoints:** For creating, updating, deleting, and retrieving pins.
* **Friendship Endpoints:** For sending, accepting, and managing friend requests, as well as retrieving friends lists.

**9. Example Technologies and Libraries**

* **Frontend:**
  + React.js or Vue.js for building a dynamic UI.
  + Leaflet or Google Maps API for map functionalities.
  + Fabric.js or Konva for custom overlays on a canvas.
* **Backend:**
  + Express.js (for Node.js) or Django (for Python) for building the backend API.
  + Sequelize (for Node.js) or SQLAlchemy (for Python) for ORM and database interaction.
* **Database:**
  + PostgreSQL or MySQL for SQL databases.
  + MongoDB for NoSQL option.

**Example Workflow**

1. **User Registration/Login:**
   * Users create accounts or log in.
   * Authentication is managed by the backend.
2. **Map Interaction:**
   * Users view the map and place pins.
   * Pins are saved to the backend with location data.
3. **Profile Management:**
   * Users can create and edit profiles, and manage profile pictures and personal details.
4. **Friendship Management:**
   * Users can send, receive, and manage friend requests.
   * The system tracks and displays friendships and connections.

**Basic Example Code Snippets**

**Frontend: Adding Pins with Leaflet**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Map Pin App</title>

<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />

<style>

#map { height: 600px; }

</style>

</head>

<body>

<div id="map"></div>

<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>

<script>

const map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

maxZoom: 19,

}).addTo(map);

map.on('click', function(e) {

L.marker([e.latlng.lat, e.latlng.lng]).addTo(map)

.bindPopup('A new pin!')

.openPopup();

});

</script>

</body>

</html>

This example uses Leaflet to display a map and allow users to place pins by clicking on the map. For a complete application, you would integrate this with your backend to manage users, profiles, and friendships.

**Security and Best Practices**

* **Data Security:** Implement security measures to protect user data, including encryption, secure authentication, and validation.
* **Scalability:** Design your system to handle increasing numbers of users and interactions.
* **Testing:** Thoroughly test both frontend and backend components to ensure reliability and a smooth user experience.

By following these guidelines, you’ll be able to build a web application with interactive maps, user profiles, and social features.

Top of Form

Bottom of Form