Summative Project: Event Locator App - Backend

Start Assignment

- Due Thursday by 11:59pm
- Points 100
- Submitting a website url or a file upload

Project Overview

This project aims to evaluate your backend development skills using Node.js and related technologies. You will build a multi-user event locator application, demonstrating your ability to handle geographical data, user preferences, asynchronous tasks, internationalization, and robust testing.

Learning Objectives

- Demonstrate proficiency in building backend applications with Node.js.
- Apply knowledge of database design and interaction, including geospatial data handling.
- Implement asynchronous task handling using a message queuing system.
- Develop internationalization features to support multiple languages.
- Write comprehensive unit tests to ensure code quality and reliability.

Project Description

Develop a multi-user event locator application allowing users to discover events based on location and preferences. The application should include:

- **User Management:** Secure user registration and login with password hashing. Users should be able to set their location and preferred event categories.
- **Event Management:** Users (or administrators) can create, read, update, and delete events, including event details, location (latitude/longitude), date/time, and categories.
- **Location-Based Search:** Implement a search functionality that allows users to find events within a specified radius of their location.
- **Category Filtering:** Enable users to filter events based on categories.
- **Multilingual Support (i18n):** Enable users to select their preferred language for the user interface.
- **Notification System (Queuing):** Utilize a message queue (e.g., Redis Pub/Sub, RabbitMQ) to send notifications to users about upcoming events that match their preferences (optional: include a delay to send notifications closer to the event).
- **Unit Testing:** Write unit tests for core functionalities, including user authentication, event management, location-based search, and the notification system.

Additional Features

- Implement event ratings and reviews.
- Integrate with a mapping service (e.g., Google Maps API) to display event locations on a map.
- Add a feature to allow users to save favorite events.
- Implement real-time updates for event changes.

Technical Considerations

- **Databases:** Choose a relational database (e.g., PostgreSQL with PostGIS) to store user data, event data, and location information.
- **Queuing System:** Use Redis Pub/Sub or RabbitMQ to manage asynchronous notifications.
- **Node.js Framework:** Utilize Express.js or a similar framework to structure your application.
- **Authentication:** Implement secure password hashing (e.g., bcrypt) and consider using Passport.js for authentication.
- **i18n Libraries:** Choose an i18n library (e.g., i18next) for internationalization.
- **Testing Framework:** Use Jest or Mocha for unit testing.
- **Geospatial libraries:** if using node postgres with postgis, utilize the built-in geospatial functions.

Project Deliverables

- **Functional Backend Application:** A working Node.js application meeting all requirements.
- **Database Schema:** A well-structured/Normalized Database Schema
- **Source Code:** Well-organized, commented code adhering to coding standards.
- **Unit Tests:** A comprehensive suite of unit tests.
- **Project Documentation:** Clear documentation outlining project setup, technical choices, and usage instructions.
- **Video Presentation and Demonstration**
  - Everyone will create a video presentation that lasts at most 5 minutes to showcase their project. The video should:
  - Explain the Project: Clearly describe the purpose and functionality of the event locator application.
  - Demonstrate Key Features: Showcase the core features, including user registration/login, event creation/search, multilingual support, and the notification system.
  - Highlight Technical Choices: Discuss the technologies used (databases, queuing system, i18n library, etc.) and explain the reasons behind those choices, especially related to geospatial data.
  - Address Challenges and Solutions: Describe any challenges faced during development and the solutions implemented to overcome them.
  - Showcase Code and Tests: Briefly present segments of the codebase, highlighting key functionalities and unit tests.

Assessment Criteria

- **Functionality (50%):**
  - User registration/login and preference setting (10 points)
  - Event CRUD operations and location handling (20 points)
  - Location-based search and category filtering (10 points)

    - Multilingual support (10 points)
- **Database schema(10 %):** Well normalized Database Schema
- **Implementation of Technologies (10%):** Effective use of Node.js, chosen database (especially geospatial features), queuing system, and i18n.
- **Code Quality and Best Practices (15%):** Adherence to coding standards, use of design patterns, and error handling.
- **Video Presentation and Demonstration (15%):**
    - Clarity and Completeness: The video effectively explains the project, demonstrates key features, and covers all required aspects.
    - Technical Accuracy: The information presented is technically accurate and demonstrates a clear understanding of the concepts.
    - Presentation Quality: The video is well-organized, engaging, and professionally presented with good audio and visuals.

Rubric for Multi-User File Manager Application

Criteria	Ratings				Pts
Functionality	**50 to >40.0 pts Exemplary** All features are fully implemented and function seamlessly, including user management, file operations, multilingual support, and queuing system.	**40 to >30.0 pts Proficient** Most features are implemented and functional, with minor limitations or errors.	**30 to >20.0 pts Developing** Some features are implemented with significant limitations or errors.	**20 to >0 pts Needs Improvement** Few features are implemented, or they are largely non-functional.	50 pts
Implementation of Technologies	**20 to >15.0 pts Exemplary** Effective and appropriate use of Node.js, Redis, database, i18n library, and queuing system.	**15 to >10.0 pts Proficient** Generally appropriate use of technologies with minor inconsistencies or inefficiencies.	**10 to >5.0 pts Developing** Some technologies are used inappropriately or inefficiently.	**5 to >0 pts Needs Improvement** Technologies are poorly utilized or integrated.	20 pts
Code Quality and Best Practices	**15 to >10.0 pts Exemplary** Code is exceptionally clean, well-organized, and efficient, adhering to best practices and coding standards.	**10 to >8.0 pts Proficient** Code is generally clean and organized, with minor areas for improvement.	**8 to >3.0 pts Developing** Code lacks organization or efficiency, hindering readability and maintainability.	**3 to >0 pts Needs Improvement** Code is poorly written, disorganized, and inefficient.	15 pts
Testing	**10 to >7.0 pts Exemplary** Comprehensive unit tests cover all core functionalities, ensuring code quality and reliability.	**7 to >5.0 pts Good** Unit tests cover most core functionalities with some gaps in coverage.	**5 to >2.0 pts Proficient** Unit tests are minimal or inadequate, leaving significant portions of the code untested.	**2 to >0 pts Needs Improvement** Unit tests are missing or non-functional.	10 pts
Presentation	**5 to >3.0 pts Exemplary** The video presentation is clear, complete, and engaging, effectively showcasing the project and demonstrating a strong	**3 to >2.0 pts Proficient** The presentation is informative but may have minor gaps or inconsistencies in clarity or completeness.	**2 to >1.0 pts Developing** The presentation lacks clarity, completeness, or technical accuracy.	**1 to >0 pts Needs Improvement** The presentation is poorly organized, incomplete, or fails to demonstrate understanding of the project.	5 pts
					Total Points: 100

understanding of the concepts.