International Institute of Information Technology Bangalore

SE Project Report: CampQuest

Team Name: Team404

Team Members:

1. Kartikeya Dimri – IMT2023126
2. Ayush Mishra – IMT2023129
3. Harsh Sinha – IMT2023571
4. Santhosh Vodnala – IMT2023622

Github Repository Link:
https://github.com/Ayush-Mishra-0018/CampQuest-SE_Project/

December 7, 2025

Contents

Note

A detailed project description, repository structure, and extended testing documentation are available in the README file hosted in the GitHub repository linked above. This report summarizes and formalizes the project work as per Software Engineering life-cycle phases.

1 Introduction

CampQuest is a full-stack web application that allows users to discover, create, and review campgrounds. The project is inspired by YelpCamp and was developed to demonstrate the end-to-end Software Development Life Cycle (SDLC) as part of the Software Engineering course.

The system supports authenticated content creation and modification while allowing unauthenticated users to browse campgrounds and reviews. The application emphasizes secure authentication, ownership-based authorization, server-side validation, and robust error handling.

2 CampQuest Features

The major features implemented in CampQuest are:

- User registration, login, and logout.
- Session-based authentication and authorization.
- CRUD operations for campground listings.
- Review creation and deletion for campgrounds.
- Ownership-based access control.
- Server-side validation using Joi schemas.
- Centralized error handling using custom error classes.
- Responsive UI using EJS templates and Bootstrap.

3 Requirement Analysis Phase

The Requirement Analysis phase resulted in a formal Software Requirements Specification (SRS) document that defines both functional and non-functional requirements for CampQuest.

3.1 Functional Requirements

- User authentication and session management.
- Campground management with full CRUD functionality.
- Review creation and deletion with proper ownership checks.
- Server-side validation and authorization enforcement.

3.2 Non-Functional Requirements

- Security through password hashing and protected routes.
- Maintainability via modular MVC-based architecture.
- Usability through responsive and intuitive UI design.
- Reliability using centralized error handling mechanisms.

The complete specification is documented in the SRS prepared during this phase :contentReference[oaicite:0]index=0.

4 Design Phase

The design phase focused on modeling system behavior, structure, and interactions using standard UML and ER diagrams.

4.1 Activity Flow Diagram

Represents high-level user interaction flows such as authentication, campground creation, and review workflows.

4.2 Class Diagram

Defines core entities including User, Campground, and Review along with their attributes and associations.

4.3 ER Diagram

Illustrates database relationships, including one-to-many relations between Campgrounds and Reviews and user ownership constraints.

4.4 Object Diagram

Shows runtime object instances and their relationships during system execution.

4.5 Sequence Diagram

Models interaction sequences between client, server, controllers, models, and database for key operations.

4.6 Use Case Diagram

Captures system functionality from the perspective of authenticated and unauthenticated users.

5 Construction Phase

The system was implemented using Node.js and Express.js, with MongoDB used for persistent storage. EJS templates were used for server-side rendering, and Bootstrap was used for UI styling.

The codebase follows a modular structure separating concerns into models, controllers, routes, middleware, and views. This structure improves maintainability and testability.

6 Testing Phase

Testing was carried out extensively using Jest, covering both unit and integration scenarios.

6.1 Unit Testing

Unit tests validate:

- Custom helper utilities such as ExpressError and async wrappers.
- Mongoose models for schema validation and constraints.
- Middleware logic for authentication and authorization.
- Client-side form validation behavior.

6.2 Integration Testing

Integration tests verify complete workflows using MongoDB Memory Server:

- User registration, login, session persistence, and logout.
- Campground creation, viewing, editing, and deletion.
- Review creation, aggregation, and deletion.
- Cascade deletion of reviews when a campground is deleted.
- Multi-user authorization and access control enforcement.

Each test suite runs in isolation with full setup and teardown to ensure reliability and data integrity.

7 Project Setup and Execution

7.1 Cloning the Repository

```
git clone https://github.com/Ayush-Mishra-0018/CampQuest-SE_Project.git
cd CampQuest-SE_Project
```

7.2 Installing Dependencies

```
npm install
```

7.3 Environment Setup

Create a `.env` file in the project root directory with the following variable:

```
MONGO_URI=mongodb://127.0.0.1:23017/YELPCAMP
```

Ensure a MongoDB instance is running and accessible at the above address.

7.4 Seeding the Database (Optional)

```
node Seeds/seed.js
```

7.5 Running the Application

```
node index.js
```

The application can be accessed at `http://localhost:3000`.

7.6 Running Tests

```
# Run all tests
npm test

# Run unit tests only
npm run test:unit

# Run integration tests only
npm run test:integration

# Run a specific test file
npx jest tests/integration/user.lifecycle.test.js
```