International Institute of Information Technology Bangalore

Software Requirements Specification

for

CampQuest

Version 2.0

Prepared by:

Team404

International Institute of Information Technology, Bangalore

Kartikeya Dimri (IMT2023126)
Ayush Mishra (IMT2023129)
Harsh Sinha (IMT2023571)
Santhosh Vodnala (IMT2023622)

November 12, 2025

Software Engineering (CSE-303) – Project Work

Contents

Revision History

Name	Date	Reason For Changes	Version
Team404	November 05, 2025	Initial draft	1.0
Team404	November 12, 2025	Modified some features	2.0

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for the **CampQuest** web application, Version 2.0. This product is a full-stack campground listing application inspired by YelpCamp. The scope of this document covers all features related to campground management, review management, and user authentication.

1.2 Document Conventions

This document follows a standard IEEE-style template. Functional requirements are uniquely identified with a prefix (e.g., **REQ-1**). Priorities for features are defined as **High**, **Medium**, or **Low**. The term "user" refers to an end-user of the software, and "system" refers to the CampQuest application itself.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- **Developers:** To understand what to build.
- **Project Managers:** To manage the project scope and plan.
- **Testers (QA):** To create test cases and verify functionality.
- **Stakeholders:** To review the product's capabilities.

It is suggested to first read Sections 1 and 2 for a high-level overview. Section 4 contains the detailed system features, which are the core of the document.

1.4 Product Scope

CampQuest is a full-stack web application that allows users to find, create, and review campgrounds. The system's purpose is to provide a platform for users to share their campground experiences. It supports complete **CRUD** (Create, Read, Update, Delete) operations for campgrounds as well as reviews and allows users to add, see, edit and delete reviews for those campgrounds. All content creation and modification are protected by user authentication and authorization.

1.5 References

- **Udemy Course:** `https://www.udemy.com/course/the-web-developer-bootcamp/`
- **IEEE Std. 830-1998:** Recommended Practice for Software Requirements Specifications.

2 Overall Description

2.1 Product Perspective

CampQuest is a self-contained full-stack web application. It is not a follow-up or component of a larger system. It functions as a monolithic web application built on Node.js runtime using Express.js framework, interfacing with a MongoDB database for data persistence. It serves server-rendered HTML pages (via EJS) to the user's web browser.

2.2 Product Functions

The major functions the product must perform are the following.

- User registration, login, and logout (Authentication).
- Authorization to ensure users only modify their own content.
- Full CRUD operations (Create, Read, Update, Delete) for campgrounds. Update and Delete operations are only applied on user owned campgrounds.
- Create and Read operations for reviews associated with campgrounds. Delete operations for user owned reviews associated for user owned campgrounds.
- Server-side validation and error handling for all user inputs.

2.3 User Classes and Characteristics

Unauthenticated User (Guest): This user class has not logged in. They have read-only access and can browse and view all campgrounds and their associated reviews.

Authenticated User (Member): This user class has successfully registered and logged in. In addition to guest privileges, a member can:

- Create new campground listings.
- Update or delete their **own** campground listings.
- Add reviews to any campground.
- Delete their **own** reviews.

2.4 Operating Environment

The software will operate in the following environment:

- **Server-Side:** Node.js runtime environment with the Express.js web framework. Requires a connection to a MongoDB database instance.
- **Client-Side:** A modern web browser (e.g., Chrome, Firefox, Safari, Edge) that supports HTML5, CSS3 (Bootstrap), and JavaScript.

2.5 Design and Implementation Constraints

The developers are advised to use industry standard technologies and frameworks. The recommended technology stack decisions are as follows:

- **Backend:** Node.js runtime and Express.js web framework
- **Database:** MongoDB with Mongoose as the Object Data Modeler (ODM).

- **Templating Engine:** EJS (Embedded JavaScript).
- **Styling:** Bootstrap frontend toolkit. Custom CSS can be used in certain circumstances.
- **Version Control:** Git version control. GitHub hosting for collaboration.

2.6 User Documentation

No formal user manuals or tutorials will be delivered with this version. All user guidance will be provided directly within the web interface (e.g., form labels, intuitive navigation).

2.7 Assumptions and Dependencies

- The system assumes all users have a stable internet connection.
- The system depends on the continuous availability and accessibility of the MongoDB database service.
- The system depends on third-party NPM packages (e.g., Express, Mongoose, EJS) being available and functional.

3 External Interface Requirements

3.1 User Interfaces

The application shall provide a web-based Graphical User Interface (GUI) accessible via a standard web browser.

- The UI shall be rendered server-side using EJS templates.
- The UI shall be styled using the Bootstrap framework and custom CSS for a responsive, mobile-first design.
- The UI shall provide clear navigation (e.g., a navigation bar).
- Error messages from server-side validation shall be displayed to the user on the relevant forms.

3.2 Hardware Interfaces

Not applicable. The CampQuest software does not interface directly with any specialized hardware components.

3.3 Software Interfaces

The system shall interface with the following external software components:

- **Node.js Backend:** The application shall use Node.js as the runtime environment to run JavaScript on the server side. It will serve as the backend platform for handling API requests, connecting to MongoDB using Mongoose, and managing CRUD operations for Users, Campgrounds, and Reviews.
- **MongoDB Database:** The application shall use the Mongoose library to connect to, create, read, update, and delete data (Users, Campgrounds, Reviews) from a MongoDB database.

3.4 Communications Interfaces

- The application shall communicate with the user's client (web browser) using the HTTP/HTTPS protocol.
- All data transfer between the client and server will be in the form of standard HTTP requests and responses.

4 System Features

This section defines the functional requirements of the system, organized by major feature.

4.1 Feature 1: User Authentication

4.1.1 Description and Priority

Provides mechanisms for user registration, login, and logout to manage user identity. **Priority: High.**

4.1.2 Stimulus/Response Sequences

- **Register:** User submits the registration form. System validates data, creates a new user, logs them in, and redirects to the campgrounds index.
- **Login:** User submits the login form. System validates credentials, creates a user session, and redirects to the campgrounds index.
- **Logout:** User clicks the 'Logout' link. System destroys the user session and redirects to the home page.

4.1.3 Functional Requirements

- **REQ-1:** The system shall provide a registration page for new users (e.g., username, password).
- **REQ-2:** The system shall provide a login page for existing users.
- **REQ-3:** The system shall provide a logout mechanism that terminates the user's session.
- **REQ-4:** The system shall restrict access to creation and modification functions to authenticated users.

4.2 Feature 2: Campground Management (CRUD)

4.2.1 Description and Priority

Allows users to create, read, update, and delete campground listings. **Priority: High.**

4.2.2 Stimulus/Response Sequences

- **Index (Read):** User navigates to the '/campgrounds' page. System displays a list of all campgrounds.
- **Show (Read):** User clicks on a specific campground. System displays the detailed page for that single campground, including its reviews.
- **Create:** Authenticated user submits the 'New Campground' form. System validates data, creates the campground, and redirects to the new campground's 'Show' page.
- **Update:** The owner of a campground submits the 'Edit Campground' form. System validates data, updates the campground, and redirects to the updated 'Show' page.
- **Delete:** The owner of a campground triggers the delete action. System deletes the campground and its associated reviews, then redirects to the 'Index' page.

4.2.3 Functional Requirements

- **REQ-5:** All users (guest and authenticated) shall be able to view a list of all campgrounds.
- **REQ-6:** All users shall be able to view the details of a single campground.
- **REQ-7:** Only authenticated users shall be able to create a new campground listing.
- **REQ-8:** Only the authenticated user who created a campground (the owner) shall be able to update its details (Authorization).
- **REQ-9:** Only the authenticated user who created a campground (the owner) shall be able to delete it (Authorization).
- **REQ-10:** The system shall perform server-side validation on all data submitted for campground creation or updates.

4.3 Feature 3: Review Management

4.3.1 Description and Priority

Allows authenticated users to add and delete reviews for existing campgrounds. **Priority: High.**

4.3.2 Stimulus/Response Sequences

- **Create:** Authenticated user submits the 'New Review' form on a campground's 'Show' page. System validates data, creates the review, associates it with the campground, and refreshes the 'Show' page.
- **Delete:** The owner of a review triggers the delete action for that review. System deletes the review and refreshes the 'Show' page.

4.3.3 Functional Requirements

- **REQ-11:** Only authenticated users shall be able to add a review to a campground.
- **REQ-12:** Only the authenticated user who created a review (the owner) shall be able to delete it (Authorization).
- **REQ-13:** The system shall perform server-side validation on all review submissions.
- **REQ-14:** All reviews for a campground shall be displayed on that campground's 'Show' page.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- TBD: Specific page load times (e.g., for the campground index) under normal conditions.
- TBD: Server response time for API/form submissions should be under a specified limit (e.g. less than 2 seconds).

5.2 Safety Requirements

Not applicable. The CampQuest application is a data-listing website and has no safety-critical functions.

5.3 Security Requirements

- **REQ-15:** All user passwords must be securely hashed using a one-way algorithm (e.g., bcrypt) before being stored in the database. Passwords must never be stored in plain text.
- **REQ-16:** The system must enforce authorization rules on the server side, not just by hiding UI elements. (e.g., A user must not be able to delete another user's campground by guessing the URL).

5.4 Software Quality Attributes

- **Maintainability:** The application logic shall be separated (e.g., models, views, controllers/routers) to ensure modularity and ease of maintenance.
- **Usability:** The interface shall be clean, intuitive, and responsive, following standards set by the Bootstrap framework.
- **Reliability:** The system must include basic error handling to gracefully manage common issues (e.g., database connection failure, invalid user input) without crashing.

5.5 Business Rules

- **BR-1:** A user must be authenticated to create any content (campgrounds or reviews).
- **BR-2:** A user can only modify or delete content that they are the author of.
- **BR-3:** A review must be associated with one (and only one) campground.
- **BR-4:** A campground can have many reviews.

6 Other Requirements

This section details requirements planned for future versions.

- **REQ-F1:** The system shall provide functionality for users to directly upload one or more images for their campground listings via the user interface.
- **REQ-F2:** The system shall eventually provide search and filter functionality for the campground index page.

A Appendix A: Glossary

CRUD Create, Read, Update, Delete. The four basic functions of persistent storage.

EJS Embedded JavaScript. A server-side templating engine that generates HTML with plain JavaScript.

Mongoose An Object Data Modeler (ODM) library for Node.js and MongoDB.

Node.js A JavaScript runtime environment that executes JavaScript code outside a web browser.

SRS Software Requirements Specification. This document.

TBD To Be Determined. Indicates that information is not yet available.

B Appendix B: Analysis Models

This section is reserved for analysis models, such as Use Case diagrams, Class diagrams, or State-Transition diagrams, which may be developed in a later phase to further illustrate the requirements.

C Appendix C: To Be Determined List

The following items are TBD and must be resolved:

1. (from 5.1) Specific performance metrics for page load times.
2. (from 5.1) Specific performance metrics for server response times.