Software Requirements Specification

for

FindMyCoolie

Version 1.0 approved

Prepared by Ranjan Kumar

Heritage Institute of Technology

<date created>

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
FindMyCoolie			1.0

1. Introduction

1.1 Purpose

Meet FindMyCoolie, a game-changing web app that makes your train travel very easy or effortless. Imagine one click connecting you with a reliable coolie who'll take care of your heavy luggage. No more struggling with bags during your journey – FindMyCoolie is here to simplify it all. Just click, connect, and enjoy a stress-free ride with the help of a dedicated coolie. Travel lighter, travel smarter with FindMyCoolie!.

1.2 Document Conventions

The document is created by keeping in mind the IEEE template for system requirement. The system features are written based on the functional point of view.

1.3 Intended Audience and Reading Suggestions

Intended Audience

Users can connect to reliable coolies from train at their fingertips who can assist you in carrying bags..

Reading Suggestions

- This SRS contains every major and minor details of the Application.
- Users should read beginning with the overview sections and proceeding through the System features to understand the proper way of using the app.
- Developers should read beginning with the overview sections and proceeding through the overall description, system features, non-functional features.

1.4 Product Scope

Navigating busy train stations with heavy luggage can be a challenging task, but with FindMyCoolie, we've simplified the process for users. Our user-friendly web app connects users with reliable coolies (porters) who are ready to assist users in carrying bags of him, ensuring a hassle-free journey. Simply input your travel details, and our app will match you with available coolies at your departure and arrival stations. Users can ay goodbye to the stress of managing luggage on your own

By joining our platform, coolie will gain access to a network of travelers seeking assistance with their luggage at train stations. This means more job opportunities and a chance to showcase your skills in providing excellent service. FindMyCoolie not only connects coolies with potential clients but also allows him to build a reputation through user reviews, establishing trust and credibility. Coolie can take control of schedule, set rates, and enjoy the flexibility of working when and where they want.

1.5 References

None

2. Overall Description

2.1 Product Perspective

FindMyCoolie is a standalone web application designed to revolutionize the way train travelers connect with coolies at various train stations. This product is an innovative solution, introducing a new and self-contained service to streamline luggage assistance during train journeys. While it may integrate with external mapping and location APIs for enhanced functionality, FindMyCoolie operates independently, ensuring a user-friendly and efficient experience. The system comprises major components such as user registration, a matching algorithm, scheduling, and a review system. The context of this product revolves around providing a seamless, on-demand connection between users and coolies, transforming the travel experience.

2.2 Product Functions

- **Language Selection:** Users will have the flexibility to choose their preferred language for the platform's interface.
- **User Registration and Authentication / Login:** Allows users to register on FindMyCoolie by providing necessary details such as name, email, and password.
- **Trip Details Input:** Empowers users to input their travel details, including departure and arrival stations.
- **Real-time Location Tracking:** Provides a real-time tracking feature for users to monitor the location of their assigned coolie.
- **Ratings and Reviews System:** Implements a feedback system where users can rate and review coolie services.
- **Payment:** Allows users to make payments for the coolie services seamlessly through integrated payment gateways
- **View Coolie Profile:** Allows users to view detailed information about a coolie
- **Request Coolie Assistance:** Enables users to request assistance from a coolie based on their travel details
- **In-app Messaging:** Facilitates communication between users and coolies through an in-app messaging feature.
- **Review and Feedback:** Allows users to provide feedback on the coolie's services.

- **Settings:** Provides users with the ability to customize preferences, notifications, or any other settings related to their FindMyCoolie experience.
- **Log Out:** Allows users to securely log out from the system.
- **Customer Support:** Users can access customer support services in case of any issues
- **User Profile:** Allows users to keep track of their historical interactions, ratings, and reviews, providing a comprehensive view of their FindMyCoolie journey.

2.3 User Classes and Characteristics

The different user classes involved in the software are:

Sl. no	User Classes	Characteristics
1.	Travelers	Regular users seeking luggage assistance during train journeys and utilize the platform to connect with coolies for efficient and convenient travel.
2.	Coolies	Service providers offering assistance with luggage at train stations, Use the platform to connect with travelers and showcase their skills.
3.	Administrator	Administers the platform to maintain authenticity and trust. Verifies user and coolie profiles for legitimacy, Resolves disputes and ensures compliance with platform policies, Monitors and enhances overall system security, Manages and oversees user and coolie verification processes, Collaborates with customer support to address escalated issues.

2.4 Operating Environment

The software will operate in a web-based environment, compatible with commonly used browsers. It requires stable internet connectivity for both users and coolies to access and utilize the platform effectively.

2.5 Design and Implementation Constraints

When we will design and implementing our concept of FindMyCoolie then we have to design frontend as well as backend part. In the frontend part we will use HTML, CSS, Javascript,

Bootstrap, Tailwind css, Sass, Reactjs and for backend we will use Nodejs and Express.js and for database we will use MongoDB and Constraints include adherence to security measures, compatibility with specific APIs for mapping and location services, and reliance on external technologies for seamless operation.

2.6 User Documentation

User documentation, including manuals, on-line help, and tutorials, will be delivered with the software. The format and standards for documentation will adhere to industry best practices

2.7 Assumptions and Dependencies

Assumptions include users and coolies having internet-enabled devices and coolies being responsible for their transportation. Dependencies involve reliance on external APIs for accurate location and mapping data.

3. External Interface Requirements

3.1 User Interfaces

Logical Characteristics: The user interface will feature a user-friendly design, including input fields for trip details, real-time coolie tracking, and an interactive map. It should be consistent with industry standards for ease of use.

Standard Buttons and Functions: The interface will incorporate standard buttons for actions such as requesting assistance, viewing coolie profiles, and providing feedback. Consistency with industry-standard UI elements will be maintained.

3.2 Hardware Interfaces

• Devices: Phone or Computer.
• Processor: Pentium or higher.
• RAM: 100MB or higher.
• A proper internet connection of more than 1kbps.

3.3 Software Interfaces

• Operating System: Linux, windows, Mac etc.
• Tools for Development: HTML, ReactJS, Express.JS, Node.JS, MongoDB etc.
• Programming languages: Javascript.
• Since the application will also be web-based, a proper server is required to host the Web-application.
• In addition to all the above mentioned software interfaces, separate software can be required for application frameworks, SMS and email servers and much more to come.

3.4 Communications Interfaces

In-App Messaging: The platform will feature an in-app messaging system to facilitate communication between users and coolies. It will adhere to industry standards for messaging interfaces.

Notifications: Users and coolies will receive notifications for matched requests, coolie arrivals, and other important updates. Standard communication protocols will be used for timely and reliable notifications.

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

4.2 System Feature 2 (and so on)

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>