Software Requirements Specification (SRS)

DIL MANGE MORE

January 28, 2024

Names:
Aditya Singh 2101020
Jashanpreet 2101090
Kartik Kumar 2101096

Index

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to serve as a detailed roadmap for the development of the ’Dil Mange More’ food delivery web application, leveraging the power of the MERN stack.We will use Prototype model. Beyond a mere listing of requirements, it comprehensively outlines both functional and non-functional aspects, design constraints, and other vital considerations, ensuring a methodical and triumphant development journey. This application aims to empower users with the convenience of ordering food online, offering choices like home delivery, pickups, and seamless exploration of nearby restaurants. ’Dil Mange More’ brings the entire culinary experience to the user’s doorstep, promising a delightful and stress-free way to savor favorite meals from the comfort of home.

1.2 Document Conventions

In maintaining top-notch coding standards, we adhere to industry best practices, embracing the Airbnb JavaScript Style Guide for React.js components, adopting ES6 syntax, and enforcing consistent naming conventions. To further enhance code clarity and maintainability, in-code documentation leverages JSDoc. Additionally, the ER diagram is employed to illuminate the database structure, providing a clear visual representation that aids developers in comprehending the intricacies of data storage and retrieval.

1.3 Intended Audience and Reading Suggestions

Designed for a diverse audience, this document caters to developers, testers, project managers, and stakeholders actively engaged in the ’Dil Mange More’ project. Developers find technical specifications to guide implementation, project managers delve into project scope, milestones, and challenges, while stakeholders gain valuable insights into the application’s technical nuances. By catering to varied roles, this document facilitates a collaborative and informed approach to the project.

1.4 Project Scope

The ’Dil Mange More’ project ambitiously encompasses the development of a user-centric food delivery web application, employing the MERN stack for optimal performance. Within this scope, key components include a responsive React.js frontend, a scalable Node.js and Express.js backend, and MongoDB for robust data storage. Embracing a holistic approach, the project entails not only secure user authentication and real-time order processing but also seamless integration with external payment gateways. The envisioned scope promises a feature-rich and user-friendly platform that redefines the online food ordering experience.

1.5 References

For guidance and best practices, refer to [MongoDB Documentation](https://docs.mongodb.com/), [Express.js Documentation](https://expressjs.com/), [React.js Documentation](https://reactjs.org/docs/getting-started.html), and [Node.js Documentation](https://nodejs.org/en/docs/). These resources provide invaluable support in harnessing the capabilities of the MERN stack and ensuring the successful implementation of the ’Dil Mange More’ web application.

2 Overall Description

2.1 Product Perspective

The ’Dil Mange More’ web app is a standalone system that interfaces with external payment gateways and utilizes external APIs for location and order tracking. It operates independently, providing users with a seamless food ordering and delivery experience.It outlines the functional and non-functional requirements, design constraints, and other essential aspects to ensure a systematic and successful development process.This web application will allow users to order food online giving facilities like home delivery, pickups or surffing through the nearby restaurents.This app gave user the freedom to order what they want from the comfort of the home.

2.2 Product Functions

The major functions of the ’Dil Mange More’ application include user registration, food browsing, order placement, real-time order tracking, and payment processing. These functions collectively contribute to an efficient and user-friendly food delivery platform.

2.3 User Classes and Characteristics

The application caters to three main user classes: customers, administrators, and delivery personnel. Each user class possesses distinct roles and characteristics. Customers interact with the app to place and track orders, administrators manage the platform, and delivery personnel fulfill orders.

2.4 Operating Environment

The ’Dil Mange More’ software operates in a web browser environment and is compatible with major browsers such as Chrome, Firefox, and Safari. Users can access the application from various devices with internet connectivity.

2.5 Design and Implementation Constraints

The ’Dil Mange More’ app is meticulously developed in adherence to industry-standard coding practices and conventions, with a specific focus on the MERN (MongoDB, Express.js, React.js, Node.js) stack. These constraints serve as essential guidelines, influencing development decisions and fostering a codebase characterized by consistency, maintainability, and scalability.

2.6 User Documentation

To enhance the user experience, a comprehensive user documentation package is in the works. This guide will offer clear and detailed instructions for users engaging with the ’Dil Mange More’ application. From the initial user registration to the intricacies of food ordering and real-time order tracking, this documentation aims to empower users, minimize confusion, and ensure a seamless interaction with the platform.

2.7 Assumptions and Dependencies

Integral to the ’Dil Mange More’ project are several key assumptions. A stable and reliable internet connection is assumed for the effective real-time tracking of orders. Additionally, the project depends on seamless integration with external APIs, especially for accurate location services ensuring prompt order deliveries and secure payment processing. These assumptions play a pivotal role in shaping the project’s implementation and overall functionality.

3 External Interface Requirements

3.1 User Interfaces

- **Login Page:** Implemented using React with secure JWT-based authentication. Integration of OAuth for social logins (Google, Facebook) using Passport.js.
- **Registration Form:** React-based form with extensive input validation using Formik and Yup. User data is stored in MongoDB with Mongoose for schema validation.
- **Product Page (Menu Display):** Utilizes React for dynamic rendering. Axios is used for API calls to Node.js backend for real-time menu updates.
- **Shopping Cart:** Interactive React components with Redux for global state management of cart items. Optimized for responsiveness and user-friendly interactions.
- **Order Confirmation:** Node.js with the pdfkit library for server-side PDF generation of order summaries. Nodemailer is used for automated email dispatches.

3.2 Hardware Interfaces

- **Internet Connectivity:** Designed for compatibility with diverse internet-connected devices including laptops, smartphones, and tablets. Optimized for varying network speeds.
- **Peripheral Support:** Facilitates connection with peripherals like printers and POS systems for physical receipt generation, using browser-based APIs.

3.3 Software Interfaces

- **Operating System:** Platform-independent web application, compatible with Windows, macOS, Linux, and mobile operating systems.
- **Browser Support:** Ensures compatibility with major browsers (Chrome, Firefox, Safari, Edge) using responsive design principles and polyfills for cross-browser support.
- **Third-party APIs:** Integration with Stripe for payment processing, Google Maps API for location services, and Twilio for SMS notifications.

4 System Features

4.1 Registration and User Management

- **Description:** Comprehensive user registration and profile management system.
- **Functional Requirements:** Encrypted user credentials stored in MongoDB. Node.js backend handles user authentication, and React Router manages client-side routing for profile management features.

4.2 Ordering and Cart Management

- **Description:** Feature-rich and user-friendly ordering and cart management system.
- **Functional Requirements:** Real-time data handling using Axios for RESTful API interactions with the backend. Redux for state management across the ordering process. MongoDB aggregation pipelines for efficient order processing.

4.3 Payment and Checkout

- **Description:** Secure and flexible payment processing system.
- **Functional Requirements:** Integration with Stripe API for handling payments. Backend validation of transactions using Express middleware. Frontend confirmation and error handling implemented in React.

4.4 Logout and Session Management

- **Description:** Secure user session management and logout functionality.
- **Functional Requirements:** JWT for session management, with Node.js handling token creation and verification. React context for managing authentication states across the application.

4.5 Report Generation and Email Integration

- **Description:** Automated report generation and email integration for order confirmation.
- **Functional Requirements:** Server-side PDF generation using pdfkit in Node.js. Email dispatch using Nodemailer with SMTP configuration for delivering order summaries.

5 Nonfunctional Requirements

5.1 Performance Requirements

The system's performance is crucial to ensure a seamless user experience. The following criteria should be met:

- **Response Time:** The system must respond to user inputs within few seconds. Achieving low latency in API responses is essential for a responsive user interface. This involves optimizing backend processes, such as database queries and server-side computations, to meet the specified time constraint.
- **Throughput:** The system should scale to real time traffic. This includes both read and write operations on the database, ensuring efficient data processing. To achieve this, implement efficient data retrieval and storage mechanisms, consider indexing database fields, and optimize algorithms for data manipulation.

5.2 Safety Requirements

Ensuring the safety of user data and system operations is paramount. The following safety requirements should be met:

- **User Data Protection:** The system must ensure the protection of user data to prevent unauthorized access. Implement robust encryption mechanisms for data storage and transmission. Utilize hashing algorithms for sensitive information, and ensure compliance with data protection regulations such as GDPR.
- **Error Handling:** The system should provide clear error messages to users in case of failures to avoid confusion. Implementing comprehensive error logging and monitoring will aid in identifying and resolving issues promptly. Use tools for real-time error tracking and implement user-friendly error messages to assist users in troubleshooting.

5.3 Security Requirements

Security is of utmost importance to safeguard the system against potential threats. Key security requirements include:

- **Authentication:** Users must authenticate themselves before accessing sensitive features. Utilize industry-standard authentication protocols like JSON Web Tokens (JWT) for secure user verification. Implement multi-factor authentication for enhanced security.
- **Data Encryption:** All sensitive data transmitted over the network should be encrypted. Implement Transport Layer Security (TLS) to secure data in transit, and use encryption algorithms for data at rest. Regularly update encryption protocols to adhere to the latest security standards and algorithms.

5.4 Software Quality Attributes

Software quality attributes define the characteristics of the system that contribute to its understanding, maintainability, and overall quality. Key quality attributes of the system should include:

- **Maintainability:** The system should be designed to facilitate updates and maintenance. This involves using clean, modular code with comprehensive documentation to ensure that future developers can easily understand and modify the system.

- **Scalability:** The application should be able to handle an increasing number of users, transactions, and data without a degradation in performance. This involves using scalable architecture patterns and technologies.
- **Reliability:** The system should operate consistently under defined conditions, and in case of a failure, it should fail gracefully with minimal impact on the user experience.
- **Usability:** The user interface should be intuitive and user-friendly, allowing users to perform their tasks efficiently and without unnecessary complexity.
- **Testability:** The system should be designed to facilitate easy testing of all components to ensure functionality, performance, and security.
- **Portability:** The system should be adaptable to different environments without significant changes in the software.
- **Interoperability:** The system should be capable of interacting with other systems and exchange data in a seamless manner.

6 Updation

- **Introduction:** Updated the SDLC model in the Introduction 1.1