

| **TITLE:** Requirement **S**pecification Document |
| --- |

**AIM:** To learn and understand the way of analysing the gathered information in the previous phase for the development process and prepare requirement specification document. A concept of software engineering. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Course outcome of Experiment:**

Process of gathering requirements and converting them into specifications.

Document created will be used by both, the customer and the developer, to understand WHAT is going to be developed.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Roger Pressman, Software Engineering: A practitioners Approach, McGraw Hill, 2010 ,6th edition

2. Ian Somerville, Software Engineering , Addison Wesley,2011,9th edition

1. http://en.wikipedia.org/wiki/Software\_requirements\_specification

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

**Requirements analysis** in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. It is an early stage in the more general activity of requirements engineering which encompasses all activities concerned with eliciting, analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Conceptually, requirements analysis includes three types of activities:

* **Eliciting requirements**: the task of identifying the various types of requirements from various sources including project documentation, (e.g. the project charter or definition), business process documentation, and stakeholder interviews. This is sometimes also called requirements gathering.
* **Analysing requirements**: determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent conflicts.
* Recording requirements: Requirements may be documented in various forms, usually including a summary list and may include natural-language documents, use cases or process specifications.

New systems change the environment and relationships between people, so it is important to identify all the stakeholders, taken into account all their needs and ensure they understand the implications of the new systems. Analysts can employ several techniques to elicit the requirements from the customer. These may include the development of scenarios, the identification of use cases, the use of workplace observation or ethnography, holding interviews, or focus groups (more aptly named in this context as requirements workshops, or requirements review sessions) and creating requirements lists. Prototyping may be used to develop an example system that can be demonstrated to stakeholders. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced

Different types of Requirements

* Functional requirements
* Usability requirements
* Reliability requirements
* Performance requirements
* Security requirements

A typical SRS document template is shared subsequently. This document acts as a reference and will be used by both, the customer (for whom the software system is to be developed), and the organization which develops the solution. Typically, prepared by the development organization at the early stage of development by the professionals after interacting with the customer.

**Software Requirements Specification for:**

**Project:** AutoPilot - Car Selling Platform

**Version 1.0**

**Author:** Namit Kale - 16010122080

Vivaan Lakhani - 16010122094

Romil Lodaya - 16010122096

Aayush Mehta - 16010122110

**Organization:** K.J. Somaiya College of Engineering

**Date Created:** 23 / 07 / 2024

**Table of Contents**

**Table of Contents ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Product Scope 1

1.3 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Introduction**

**Purpose**

This document specifies the software requirements for the Rizz-Wheels platform, version 1.0. It outlines the scope of the platform focused exclusively on vehicle rental. The SRS covers functionalities for user registration, vehicle listings, detailed vehicle pages, rental transaction management, communication features, administration, and support.

**Product Scope**

The Rizz-Wheels platform, is a specialized software solution for managing and facilitating vehicle rentals. It is designed to provide users with an efficient and user-friendly interface for renting locomotives and other vehicles. The platform's primary benefits include streamlined rental processes, improved user satisfaction, and enhanced operational efficiency for rental businesses. Key objectives include automating rental transactions, simplifying vehicle management, and fostering effective user communication. By aligning with corporate goals of expanding rental offerings and enhancing customer engagement, the platform supports strategic growth and market competitiveness. For further details on the vision and scope, please refer to the separate Vision and Scope document.

**References**

1. https://www.zoomcar.com/in

**Overall Description**

**Product Perspective**

The Rizz-Wheels platform, version 1.0, is a new, self-contained software solution specifically developed for the rental of light motor vehicles (LMVs), such as cars and trucks. This platform is not an extension of an existing product family nor a replacement for any current system; rather, it introduces a novel approach to managing vehicle rentals through an integrated online interface.

The platform is designed to operate independently but is intended to interface with external systems, including payment gateways for processing transactions and authentication services for user verification. It encompasses all necessary functionalities for the rental process, such as user registration, vehicle listing management, rental transaction processing, and communication between users.

By providing these capabilities, the Rizz-Wheels platform aims to enhance the efficiency of vehicle rentals, streamline operations for rental businesses, and improve the overall user experience. The platform aligns with broader corporate goals of expanding rental services and optimizing customer engagement in the vehicle rental market.

**Product Functions**

The Rizz-Wheels platform is designed to support the rental of light motor vehicles (LMVs). The major functions the platform must perform include:

* **User Registration and Management:**
  + Create and manage user accounts.
  + Verify user identities.
  + Manage user profiles and preferences.
* **Vehicle Listing and Management:**
  + Add, update, and remove vehicle listings.
  + Display detailed vehicle information, including specifications and images.
  + Set and adjust rental rates and availability.
* **Search and Filtering:**
  + Search for vehicles based on criteria like make, model, year, and rental price.
  + Apply filters to refine search results.
* **Rental Transactions:**
  + Process rental bookings and payments securely.
  + Manage rental agreements and track rental status.
  + Handle cancellations and modifications.
* **User Communication:**
  + Enable messaging between renters and vehicle owners.
  + Send notifications for booking confirmations, reminders, and updates.
* **Administration and Moderation:**
  + Oversee platform settings, user accounts, and vehicle listings.
  + Monitor and moderate content to ensure compliance with platform policies.
* **Support and Helpdesk:**
  + Provide customer support through a ticketing system.
  + Offer FAQs and user guides for assistance.

**Operating Environment**

**Hardware Platform:**

* **Servers:** The platform will be hosted on cloud-based servers with scalable resources to handle varying loads. Recommended specifications include modern multi-core processors, sufficient RAM (e.g., 16GB or more), and SSD storage for optimal performance.
* **User Devices:** Users can access the platform through a variety of devices including desktops, laptops, tablets, and smartphones.

**Operating System:**

* **Server:** The platform will operate on server environments such as Ubuntu Linux (LTS versions) or other Linux distributions known for stability and security. Windows Server may also be used depending on specific deployment requirements.
* **Client:** The platform is accessible via major operating systems including Windows, macOS, iOS, and Android. The application will be tested for compatibility with the latest versions and popular older versions of these operating systems.

**Software Components:**

* **Web Server:** The platform will run on web servers like Apache HTTP Server or Nginx.
* **Database:** It will utilize a relational database management system (RDBMS) such as MySQL, PostgreSQL, or an equivalent to manage user data, vehicle listings, and transactions.
* **Application Server:** The backend of the platform will be developed using frameworks and technologies compatible with modern web applications (e.g., Node.js, Django, Ruby on Rails).
* **Browser Compatibility:** The platform will be compatible with the latest versions of major web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.

**Integration and Coexistence:**

* **Payment Gateways:** The platform will integrate with third-party payment gateways (e.g., Stripe, PayPal) for secure payment processing.
* **Authentication Services:** Integration with identity providers for user authentication (e.g., OAuth providers like Google or Facebook) may be implemented.
* **Email Services:** Email notifications and verifications will be managed through email service providers (e.g., SendGrid, Mailgun).

**Design and Implementation Constraints**

* **Corporate Policies:** Compliance with company security policies and data protection regulations, such as GDPR or CCPA, is mandatory. Adherence to corporate branding and design guidelines is required.
* **Regulatory Compliance:** The platform must comply with local and international laws governing vehicle rentals and online transactions, including consumer protection regulations.
* **Hardware Limitations:** The platform must be optimized to run efficiently on cloud servers with specified configurations (e.g., multi-core processors, 16GB+ RAM, SSD storage) to ensure scalable performance.
* **Interface Constraints:** Integration with third-party payment gateways (e.g., Stripe, PayPal) and authentication services (e.g., OAuth providers) must follow their APIs and security requirements.
* **Technology Stack:** Use of specified technologies and tools, including web servers (Apache HTTP Server or Nginx), databases (MySQL, PostgreSQL), and application frameworks (Node.js, Django), is required.
* **Security Considerations:** Implementation of robust security measures, such as data encryption, secure authentication, and regular vulnerability assessments, is necessary to protect user data and transactions.
* **Design Conventions:** Adherence to established design conventions and programming standards, including code documentation and version control practices, is required for maintainability and consistency.
* **Language Requirements:** Development may be constrained to specific programming languages and frameworks approved by the organization.
* **Communications Protocols:** The platform must use standard communication protocols (e.g., HTTPS) to ensure secure data transmission.

**User Documentation**

Not Applicable

**Assumptions and Dependencies**

**Assumptions:**

1. **Third-Party Services:** The platform assumes the availability and reliability of third-party services, such as payment gateways (e.g., Stripe, PayPal) and authentication providers (e.g., Google, Facebook). Any issues or changes with these services could impact functionality.
2. **User Access:** It is assumed that users will have reliable internet access and compatible devices (e.g., desktops, laptops, tablets, smartphones) to interact with the platform.
3. **Regulatory Changes:** The platform assumes stability in regulatory requirements related to vehicle rentals and online transactions. Significant changes in regulations could necessitate adjustments to the platform.
4. **Technological Stability:** The development assumes the continued support and stability of the chosen technologies, frameworks, and tools (e.g., Node.js, Django, MySQL). Major updates or deprecation of these technologies could affect development.
5. **Data Integrity:** It is assumed that data provided by users and third-party services will be accurate and reliable. Inaccurate data could impact the platform’s functionality and user experience.

**Dependencies:**

1. **External Software Components:**
   * **Payment Gateways:** Integration with payment processing services requires their APIs and documentation.
   * **Authentication Services:** Integration with OAuth providers depends on their API stability and documentation.
2. **Development Tools:**
   * **Development Environment:** The platform's development depends on the availability of development tools and environments specified for the project.
3. **Hardware and Hosting:** The platform relies on cloud infrastructure (e.g., AWS, Azure) for hosting, which must provide the required resources (e.g., CPU, memory, storage).
4. **Regulatory Compliance Tools:** The project may depend on external tools or services for compliance with data protection regulations (e.g., GDPR tools).
5. **Project Team:** Development and maintenance depend on the availability of skilled personnel with expertise in the chosen technologies and tools.

**External Interface Requirements**

**User Interfaces**

* All the users will see the same page when they enter this website. This page asks the users a username and a password.
* After being authenticated by the correct username and password, users will be redirected to their corresponding profile where they can do various activities.
* The user interface will be simple and consistent to eliminate need for user training of infrequent users, using terminology commonly understood by intended users of the system.

**NOTE:** Further specification about user details will be explored in later experiments.

**Hardware Interfaces**

* No extra hardware interfaces are needed.
* The system will use the standard hardware and data communication resources.
* This includes, but not limited to, general network connection at the server/hosting site, network server and network management tools.

**Software Interfaces**

**Technologies:**

* HTML / CSS
* JavaScript
* React
* Firebase Firestore
* VS Code
* Github
* Jest

**Communications Interfaces**

**Resources:**

* **HTTP Protocol:** This is used for communication between the web browser (client) and the web server. In the context of an online car rental platform, this protocol ensures that requests (like booking a car, checking availability, etc.) from users (via their browsers) are properly handled by the server.
* **TCP/IP Network Protocol:** This is the underlying network protocol that facilitates communication across the internet. It works in conjunction with HTTP to ensure data packets are correctly transmitted and received between the client (user’s browser) and the server.

**Database for Booking Information:**

* The platform will have a central database that stores all booking-related information. This includes details such as available cars, booking schedules, user profiles, payment information, and more.
* The database is crucial for storing and retrieving data in response to user requests. For example, when a user searches for available cars on specific dates, the system queries the database to fetch relevant information.

**HTTP Service for Server-Side Communication:**

* An HTTP service on the server side allows the platform’s application to interact with the database and other server resources. This service handles incoming requests from users (via HTTP) and processes them accordingly.
* For instance, when a user submits a booking request through the platform’s interface (typically a web page), the HTTP service receives this request, verifies it, updates the database as necessary, and responds back to the user’s browser with confirmation or any relevant information.

**Functionality and User Interaction:**

* Users interact with the platform through a web-based interface, where they can browse available cars, select dates, enter personal details, and complete bookings.
* The platform’s backend processes these interactions using server-side scripts and APIs (Application Programming Interfaces) that utilize the HTTP service for communication. This ensures that all user actions are reflected accurately in the database and that users receive timely responses.

**Security and Reliability:**

* Given the sensitive nature of personal and financial data involved (such as credit card information), the platform must implement robust security measures. This includes HTTPS (secure HTTP) to encrypt data during transmission, secure storage practices for sensitive information in the database, and authentication mechanisms to verify user identities.
* of a function that is called HTTP Service. This function allows the application to use the data retrieved by the server to fulfill the request fired by the user.

**System Features**

#### Locomotive Search and Booking

4.1.1 Description and Priority This feature allows users to search for available locomotive vehicles and book them online. Priority: High

4.1.2 Stimulus/Response Sequences

* **User Action**: User inputs search criteria (e.g., location, date, type of locomotive). **System Response**: Display list of available locomotive vehicles matching the criteria.
* **User Action**: User selects a locomotive from the list. **System Response**: Display details of the selected locomotive.
* **User Action**: User confirms the booking. **System Response**: Process payment and confirm the booking, displaying a confirmation message.

4.1.3 Functional Requirements

* **REQ-1**: The system shall allow users to search for locomotive vehicles based on location, date, and type.
* **REQ-2**: The system shall display a list of available locomotives that match the search criteria.
* **REQ-3**: The system shall show detailed information for each locomotive vehicle, including availability, pricing, and specifications.
* **REQ-4**: The system shall allow users to select a locomotive and proceed to booking.
* **REQ-5**: The system shall process user payments securely.
* **REQ-6**: The system shall send a booking confirmation to the user via email and on-screen message.
* **REQ-7**: The system shall handle invalid search inputs by displaying appropriate error messages.

#### User Account Management

4.2.1 Description and Priority This feature allows users to create and manage their accounts. Priority: Medium

4.2.2 Stimulus/Response Sequences

* **User Action**: User opts to create a new account. **System Response**: Display account registration form.
* **User Action**: User submits the registration form. **System Response**: Validate inputs and create a new user account, displaying a success message.
* **User Action**: User logs in with credentials. **System Response**: Authenticate user and grant access to account features.

4.2.3 Functional Requirements

* **REQ-1**: The system shall allow users to register by providing personal details and creating a password.
* **REQ-2**: The system shall validate user inputs during registration to ensure data integrity.
* **REQ-3**: The system shall allow users to log in using their email and password.
* **REQ-4**: The system shall authenticate users and grant access to account-specific features.
* **REQ-5**: The system shall allow users to update their account information.
* **REQ-6**: The system shall handle authentication errors by displaying appropriate error messages.

#### Payment Processing

4.3.1 Description and Priority This feature manages the processing of payments for locomotive rentals. Priority: High

4.3.2 Stimulus/Response Sequences

* **User Action**: User proceeds to payment after booking a locomotive. **System Response**: Display payment form.
* **User Action**: User submits payment details. **System Response**: Validate and process the payment, then display confirmation.

4.3.3 Functional Requirements

* **REQ-1**: The system shall provide a secure payment form for users to enter payment details.
* **REQ-2**: The system shall validate payment details before processing.
* **REQ-3**: The system shall securely process the payment through a payment gateway.
* **REQ-4**: The system shall update the booking status upon successful payment.
* **REQ-5**: The system shall send payment confirmation to the user via email and on-screen message.
* **REQ-6**: The system shall handle payment errors and display appropriate error messages.

#### Customer Support

4.4.1 Description and Priority This feature provides customer support functionalities to assist users. Priority: Medium

4.4.2 Stimulus/Response Sequences

* **User Action**: User navigates to the customer support section. **System Response**: Display options for contact methods (e.g., chat, email, phone).
* **User Action**: User selects a contact method and submits an inquiry. **System Response**: Confirm submission and display estimated response time.

4.4.3 Functional Requirements

* **REQ-1**: The system shall provide multiple customer support contact options.
* **REQ-2**: The system shall allow users to submit inquiries via a contact form.
* **REQ-3**: The system shall send an acknowledgment of the inquiry to the user.
* **REQ-4**: The system shall provide an estimated response time for customer support inquiries.
* **REQ-5**: The system shall allow users to view FAQs and support documentation.

**Other Nonfunctional Requirements**

#### Performance Requirements

* **Response Time**: The system should provide search results within 3 seconds under normal load conditions.
* **Scalability**: The system must support at least 10,000 concurrent users during peak times without performance degradation.
* **Availability**: The system should have 99.9% uptime, ensuring that it is available 24/7 with minimal downtime.
* **Throughput**: The system should be able to handle 500 transactions per second.

#### Safety Requirements

* **Data Backup**: The system should perform data backups every 24 hours to prevent data loss.
* **Error Handling**: The system must handle errors gracefully and log all error events for future analysis.
* **Regulatory Compliance**: The system must comply with local transportation regulations and safety standards.
* **User Instructions**: Clear safety instructions should be provided to users regarding vehicle operations and emergency procedures.

#### Security Requirements

* **Authentication**: The system must require users to authenticate using a secure login method.
* **Data Encryption**: All sensitive data, including user information and payment details, must be encrypted both in transit and at rest.
* **Access Control**: Different levels of access control must be implemented to restrict access based on user roles.
* **Compliance**: The system must comply with GDPR, CCPA, and other relevant data protection regulations.
* **Security Audits**: Regular security audits should be conducted to identify and address vulnerabilities.

#### Software Quality Attributes

* **Usability**: The system should have an intuitive user interface that is easy to navigate.
* **Reliability**: The system must function correctly under defined conditions and should handle failures gracefully.
* **Maintainability**: The system should be designed in a modular way to facilitate easy updates and maintenance.
* **Interoperability**: The system should be able to integrate with third-party services such as payment gateways and vehicle tracking systems.
* **Testability**: The system must be designed to allow for thorough testing of all components.
* **Adaptability**: The system should be easily adaptable to new requirements or changes in the business environment.

#### Business Rules

* **User Roles**: Only registered users can make bookings. Admins have full access to manage vehicles, bookings, and users.
* **Booking Restrictions**: Users cannot book more than one vehicle for the same time slot.
* **Cancellation Policy**: Bookings can be canceled up to 24 hours before the scheduled pick-up time with a full refund.
* **Payment Processing**: All payments must be processed before the booking is confirmed.
* **Vehicle Inspection**: Vehicles must be inspected and certified as safe before being listed for rental.

#### Other Requirements

* **Database Requirements**: The system should use a robust database management system capable of handling high transaction volumes.
* **Internationalization**: The system should support multiple languages and currencies to cater to a global user base.
* **Legal Requirements**: The system must include terms of service and privacy policies that comply with relevant laws.
* **Reuse Objectives**: The system's components should be designed for reuse in future projects or expansions.

**Appendix A: Glossary**

Not Applicable

**Appendix B: Analysis Models**

Not Applicable

**Post Laboratory Activity:**

1. You are required to prepare SRS document for any project. ( It could be the mini project you have completed in semester IV

**1. Introduction**

**1.1 Purpose** The purpose of this document is to define the Software Requirements Specification (SRS) for the "AcademicSync" project. The document will outline the features, functionalities, and constraints of the project.

**1.2 Product Scope** AcademicSync is designed to streamline the creation, sharing, and management of academic events such as tests and assignments. It aims to provide a centralized platform for students to collaborate, communicate, and stay organized.

**1.3 References**

* AcademicSync Project Report (2023-24)
* Firebase Documentation
* React Documentation

**2. Overall Description**

**2.1 Product Perspective** AcademicSync will be a web-based platform developed using React and Firebase. It will integrate with educational systems and provide a user-friendly interface for students and faculty.

**2.2 Product Functions**

* User Authentication
* Event Management
* Collaboration Tools
* Reminders and Notifications
* Announcement Section
* User Profiles
* Resource Repository

**2.3 User Classes and Characteristics**

* Students: Create and manage events, set reminders, view announcements, and collaborate with peers.
* Faculty/Admin: Post announcements, manage events, and oversee student activities.

**2.4 Operating Environment**

* Web-based application accessible via modern browsers (Chrome, Firefox, Safari).
* Mobile responsiveness for access on smartphones and tablets.

**2.5 Design and Implementation Constraints**

* Must use Firebase for backend services.
* Must be developed using React for the frontend.

**2.6 User Documentation**

* User Manual
* Online Help Section

**2.7 Assumptions and Dependencies**

* Users must have a valid educational email ID for authentication.
* Stable internet connection required for access.

**3. External Interface Requirements**

**3.1 User Interfaces**

* Intuitive web interface designed with Material-UI.
* Mobile-friendly layout for easy access on various devices.

**3.2 Hardware Interfaces**

* Devices with at least 4 GB RAM and internet connectivity.

**3.3 Software Interfaces**

* Integration with Firebase Authentication, Firestore, and Cloud Functions.
* Compatibility with modern web browsers.

**3.4 Communications Interfaces**

* HTTPS for secure communication between client and server.
* Email notifications for reminders.

**4. System Features**

**4.1 System Feature 1: Event Management**

* Create, edit, and delete academic events.
* Categorize events by courses.
* Set reminders for upcoming events.

**4.2 System Feature 2: Collaboration Tools**

* Share events with peers.
* Collaborate on group projects.
* Access shared resources.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

* The system should handle up to 1000 concurrent users.
* Response time for event creation should be under 2 seconds.

**5.2 Safety Requirements**

* Secure user authentication and authorization.
* Data encryption for stored information.

**5.3 Security Requirements**

* Use Firebase Authentication for secure login.
* Implement role-based access control.

**5.4 Software Quality Attributes**

* Usability: Easy to navigate interface.
* Reliability: Ensure 99.9% uptime.
* Maintainability: Modular code structure for easy updates.

**5.5 Business Rules**

* Only authenticated users can create and manage events.
* Admins have exclusive rights to post announcements.

**6. Other Requirements**

* Regular backups of user data.
* Periodic updates for security and feature enhancements.

**Appendix A: Glossary**

Not Applicable

**Appendix B: Analysis Models**

Not Applicable

1. Prepare questionnaire for the allotted project considering your lab instructor is the client for requirement gathering.

### Vehicle Rental E-commerce Website Requirements Gathering Questionnaire

#### General Information

1. Can you provide a brief overview of your vision for the vehicle rental e-commerce website?
2. Who will be the primary users of the website?
3. What types of vehicles will be available for rental?

#### Functional Requirements

1. What search criteria should users be able to use when looking for vehicles?
2. What key details should be displayed on the vehicle detail page?
3. What are the steps in the booking process from selecting a vehicle to confirmation?
4. What payment methods should be supported?
5. Should users be able to create and manage accounts on the website?

#### Performance Requirements

1. What are your expectations for the website’s response time and uptime?
2. What is the expected peak load in terms of concurrent users and transactions?

#### Safety and Security Requirements

1. What security measures do you expect to be implemented?
2. Are there any specific compliance requirements for data protection?

#### Software Quality Attributes

1. How important is usability and user experience for your website?
2. What are your reliability and maintenance expectations for the website?

#### Business Rules

1. What user roles do you envision, and what permissions should they have?
2. Are there any restrictions or rules for vehicle bookings and cancellations?

#### Other Requirements

1. Do you have any specific database or internationalization requirements?
2. Are there any legal or regulatory requirements the website must comply with?
3. What is your timeline and budget for the project?

#### Additional Questions

1. Are there any other specific features or functionalities you envision for the website?
2. Do you have any examples of similar websites that you like or would like to model this project after?
3. Consider following scenario: An institute is interested in developing a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (e.g. passwords) is stored in plain text.

Prepare SRS document for the same in the format discussed in the write-up.

**1. Introduction**

##### 1.1 Purpose

The purpose of this document is to define the requirements for the Library Information System (LIS) for an institute. The LIS will enable students and employees to manage book borrowing and returning with ease and facilitate library staff in handling day-to-day book transactions.

##### 1.2 Product Scope

The LIS will be a web application running on the institute's LAN, allowing members to borrow, return, and extend book borrowing periods. Non-members can browse/search books but cannot borrow or return them.

##### 1.3 References

* HTML5 specifications
* Institute's IT security policies
* Library management standards

#### 2. Overall Description

##### 2.1 Product Perspective

The LIS is a standalone web application that will function within the institute's LAN, reducing security risks and ensuring that only internal users can access the system.

##### 2.2 Product Functions

* Book borrowing and returning for members.
* Extension of borrowing period if no other reservations exist.
* New book entry and removal by the librarian.
* Online book browsing/search for all users.

##### 2.3 User Classes and Characteristics

* Students and Employees: Can borrow, return, and extend book borrowing periods.
* Library Staff: Manage book transactions.
* Librarian: Administrative control, add/remove books.
* Non-members: Browse and search books only.

##### 2.4 Operating Environment

The system will operate on the institute's LAN, accessible via modern web browsers supporting HTML5.

##### 2.5 Design and Implementation Constraints

* Must use HTML5 for the front-end.
* Ensure no confidential information is stored in plain text.
* Operate solely within the institute's LAN.

##### 2.6 User Documentation

User manuals and online help resources will be provided for all user classes, detailing system usage and functions.

##### 2.7 Assumptions and Dependencies

* Users will access the system within the institute's LAN.
* Users have basic knowledge of web browsing.

#### 3. External Interface Requirements

##### 3.1 User Interfaces

* Login Screen: For member authentication.
* Dashboard: For members to manage their book transactions.
* Admin Panel: For librarian to manage the book database.
* Search Interface: For all users to search books.

##### 3.2 Hardware Interfaces

No specific hardware interfaces are required beyond standard network-connected computers.

##### 3.3 Software Interfaces

* Database: MySQL or any other relational database to store book and user information.
* Web Server: Apache or Nginx to host the web application.

##### 3.4 Communications Interfaces

The application will communicate over the institute's LAN using standard HTTP/HTTPS protocols.

#### 4. System Features

##### 4.1 Book Borrowing and Returning

* Description and Priority: Allows members to borrow and return books. Priority: High
* Functional Requirements:
  + REQ-1: Members can borrow available books.
  + REQ-2: Members can return borrowed books.
  + REQ-3: System updates book status in real-time.

##### 4.2 Book Borrowing Extension

* Description and Priority: Allows members to extend the borrowing period if no reservations exist. Priority: Medium
* Functional Requirements:
  + REQ-1: Members can request an extension for borrowed books.
  + REQ-2: System checks for existing reservations before granting an extension.
  + REQ-3: System notifies members of the extension status.

#### 5. Other Nonfunctional Requirements

##### 5.1 Performance Requirements

* REQ-1: System should respond to user actions within 2 seconds.
* REQ-2: Handle up to 500 concurrent users without performance degradation.

##### 5.2 Safety Requirements

* REQ-1: Regular data backups to prevent data loss.
* REQ-2: Graceful error handling and logging.

##### 5.3 Security Requirements

* REQ-1: Encrypt all sensitive data in transit and at rest.
* REQ-2: Secure login mechanism for members and librarian.

##### 5.4 Software Quality Attributes

* Usability: Intuitive and easy-to-navigate interface.
* Reliability: System must function correctly under defined conditions.
* Maintainability: Modular design for easy updates.

##### 5.5 Business Rules

* REQ-1: Only valid members can borrow and return books.
* REQ-2: Librarian has the sole authority to add or remove books from the system.

#### 6. Other Requirements

* Internationalization: The system should support multiple languages if required by the institute.
* Legal Compliance: Adhere to data protection laws and institute policies.

**Appendix A: Glossary**

Not Applicable

**Appendix B: Analysis Models**

Not Applicable

**Post Lab Descriptive Questions answers must be handwritten and to be submitted BEFORE the next tern**.

1. What are different techniques to gather information for software development?

Different techniques to gather information for software development include:

* **Interviews**: Conducting face-to-face or virtual conversations with stakeholders to understand their needs and expectations.
* **Surveys and Questionnaires**: Distributing forms to collect feedback from a broad audience.
* **Workshops**: Organizing collaborative sessions with stakeholders to brainstorm and discuss requirements.
* **Observation**: Watching how users interact with existing systems to identify pain points and improvement areas.
* **Document Analysis**: Reviewing existing documentation, such as manuals, reports, and policy documents, to gather relevant information.
* **Prototyping**: Creating preliminary versions of the system to gather feedback and refine requirements.
* **User Stories and Use Cases**: Writing detailed descriptions of how users will interact with the system to uncover specific requirements.
* **Focus Groups**: Engaging a small group of users in discussions to explore their needs and preferences.
* **Brainstorming Sessions**: Facilitating open discussions to generate ideas and identify potential requirements.
* **Competitive Analysis**: Studying competitors' products to identify features and improvements for your own project.

1. List verification and validation techniques for requirements.

Verification and validation techniques for requirements include:

1. **Inspections**: Thoroughly reviewing requirement documents to identify errors and inconsistencies.
2. **Walkthroughs**: Conducting step-by-step presentations of requirements to stakeholders to ensure clarity and correctness.
3. **Peer Reviews**: Having colleagues or other team members review the requirements for completeness and accuracy.
4. **Prototyping**: Developing preliminary versions of the system to validate requirements with stakeholders through hands-on experience.
5. **Test Case Generation**: Creating test cases based on requirements to ensure they are testable and measurable.
6. **Requirements Traceability Matrix (RTM)**: Mapping requirements to their corresponding design elements, implementation, and tests to ensure all requirements are addressed.
7. **Checklists**: Using predefined lists of common errors and best practices to systematically verify requirements.
8. **Formal Methods**: Applying mathematical techniques to prove the correctness and completeness of requirements.
9. **Simulation**: Using models to simulate the system’s behavior and validate that the requirements will produce the desired outcomes.
10. **User Acceptance Testing (UAT)**: Engaging end-users to test the system against the requirements to ensure it meets their needs and expectations.