Software Requirements Specification (SRS) For Inter Hostel Cart

1. Introduction

The Inter Hostel cart is a comprehensive platform designed to facilitate seamless transactions and interactions among hostel residents within a college campus. It serves as a virtual marketplace where residents can buy, sell, or exchange goods and services, fostering a sense of community and convenience. This platform aims to enhance the hostel living experience by providing a centralized hub for residents to connect, communicate, and conduct transactions securely. By offering features such as item listings, messaging, and secure payment options, the Inter Hostel cart streamlines the process of buying and selling within the hostel community, promoting a more efficient and engaging living environment.

1.1 Purpose:

The purpose of the Inter Hostel cart is to establish an efficient platform for hostel residents within a college campuses to easily buy, sell, and exchange goods and services. By providing a centralized hub for transactions, the marketplace aims to enhance convenience and foster a sense of community among residents. Additionally, it seeks to promote sustainability by encouraging the reuse and recycling of items. Ultimately, the Inter Hostel cart strives to optimize the hostel living experience through connectivity, accessibility, and eco-conscious practices.

1.2 Scope:

The scope of the Inter Hostel Cart includes providing a user-friendly online platform for hostel residents to conduct transactions for goods and services. It encompasses features such as item listings, messaging, secure payment options, and a community forum, all tailored to enhance connectivity and convenience within the hostel community.

1.3 Document Conventions:

-Web Development (HTML, CSS, JAVASCRIPT, REACT)

-Backend (Oracle, Node.JS, EXPRESS.JS)

- Use of UML for diagrams (Use Case,).

-IEEE Standard 830 for SRS document format.

2. System Overview:

2.1 Description of the Banking Management System:

The Inter Hostel Cart is a sophisticated online platform meticulously designed to cater to the specific needs college hostel residents. This platform provides residents with a user-friendly interface, enabling effortless listing of items, secure communication through encrypted messaging channels, and diverse payment options for seamless transactions. Moreover, it fosters a vibrant sense of community by offering a dedicated forum for residents to engage in meaningful discussions and share insights. By prioritizing usability and accessibility, the Inter Hostel Cart optimizes the hostel living experience, enhancing connectivity and convenience among residents, all while promoting sustainable practices through the promotion of item reuse and recycling.

2.2 System Architecture:

The system will be built using the MERN stack, with a modular architecture to ensure scalability , maintainability and allowing for flexibility in adapting to evolving technology and regulatory requirements.

3. Functional Requirements:

3.1 User Accounts and Authentication:

1.**Account Creation :**

-Users can register for a new account online.

-Capture and verify user details, including personal information and identification documents.

2**.Authentication:**

-Implement a secure process for capturing and storing user information.

-Regularly update authentication logs for unusual patterns or potential security threats.

-Secure authentication methods such as username, passwords, biometrics(fingerprint), account lockout, device recognition and multi-factor authentication.

3. **Configure System Settings:**

System settings for the Inter Hostel Cart encompass configuring user authentication, integrating secure payment gateways, implementing encrypted messaging protocols, defining item listing parameters, and establishing moderation tools for the community forum

4. **System Maintenance:**

- Perform routine system maintenance tasks.

-Schedule regular data backups to prevent data loss in the event of failures.

5. **User Connectivity:**

User connectivity in the Inter Hostel Cart enables residents to engage seamlessly through secure messaging, ensuring efficient communication for transactions and inquiries. The platform fosters a vibrant community by facilitating interactions, promoting collaboration, and enhancing the overall hostel living experience through enhanced connectivity and convenience.

6. **Product Listing:**

Product listing in the Inter Hostel Cart empowers users to showcase items for sale or exchange with detailed descriptions and images. This feature facilitates efficient browsing and enables residents to discover and acquire desired goods and services within the hostel community, enhancing convenience and accessibility.

7. **Search and Filter:**

The search and filter functionality in the Inter Hostel Cart enables users to efficiently locate desired items by employing advanced search criteria and filtering options. This feature streamlines the browsing experience, allowing residents to refine search results based on various parameters such as price range, category, location, and item condition. By providing tailored search results, the platform enhances user satisfaction, accelerates decision-making processes, and fosters a more seamless and productive marketplace experience for hostel residents.

8. **Security Transactions:**

Inter-hostel cart transactions involve transferring goods or services between different hostel premises. Security measures for such transactions should include verifying the identity of both parties involved, ensuring the authenticity of the items being exchanged, and implementing secure payment methods. Hostel administrations may employ digital platforms or physical documentation to track transactions and prevent theft or fraud. Additionally, clear guidelines and protocols should be established to safeguard the interests of both buyers and sellers, promoting transparency and accountability in the process. Regular audits and inspections can further enhance security and mitigate risks associated with inter-hostel cart transactions.

9. **Transaction Operations:**

Inter-hostel cart transactions entail exchanging goods or services between hostel premises. Operations involve verifying identities, authenticating items, and using secure payment methods. Clear guidelines and audits ensure transparency and accountability, mitigating risks.

10. **Community Engagement:**

Inter-hostel cart initiatives can foster community engagement by promoting collaboration and interaction among residents. Hosting events like swap meets or food fairs where

individuals can showcase and exchange items from their carts encourages socialization and networking. Implementing feedback mechanisms allows residents to voice preferences and suggestions, fostering a sense of ownership and participation in cart activities. Collaborative decision-making processes for selecting cart vendors or determining operating hours ensure inclusivity and transparency. Furthermore, organizing workshops or skill-sharing sessions related to cart management or entrepreneurship enhances residents' abilities and strengthens community bonds, creating a vibrant and engaged hostel community.

11.**Feedback and Ratings :**

The Inter Hostel Cart receives glowing feedback and high ratings for its efficient service and diverse offerings. Patrons praise its convenience, cleanliness, and prompt delivery. With a wide selection of snacks, beverages, and essentials, it caters to varied preferences, ensuring customer satisfaction. The cart's affordability and friendly staff contribute to its popularity among hostel residents, earning it consistent praise and top-notch ratings across the board

.

12.**Administrative Tools:**

The Inter-Hostel Cart's administrative tools streamline operations. Its dashboard enables inventory tracking, allowing staff to monitor stock levels and replenish items efficiently. Integrated ordering systems facilitate seamless procurement, ensuring timely restocking of essential supplies. Customizable reporting features offer insights into usage trends, aiding in strategic decision-making. Communication tools facilitate collaboration among staff, enhancing coordination and productivity. User-friendly interfaces simplify tasks, enabling swift navigation and reducing training time. Overall, these administrative tools optimize the Inter-Hostel Cart's functionality, enhancing its effectiveness in serving hostel communities.

4. Non-Functional Requirements

Non-functional requirements are important because they can have a significant impact on the overall quality and success of a software system. They are often closely related to the system' performance, security, and usability. They also help to ensure that the system is maintainable, portable, and compliant with relevant laws and regulations.

Non-Functional Requirements are the constraints or the requirements imposed on the system. They specify the quality attribute of the software. Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more.

In summary, non-functional requirements are a critical aspect of software engineering that describe the qualities and characteristics of the software system beyond its functional

requirements. These requirements ensure that the software system meets the expectations of its users and stakeholders and performs efficiently and reliably under different conditions and loads.

Safety -

The Backup System is designed to facilitate the backup process of critical data within a bank's infrastructure. This system aims to provide a reliable and efficient way to create, manage, and restore backups of essential databases, files, and configurations to ensure data integrity, availability, and disaster recovery readiness.

The system shall support multiple user accounts with different access levels (admin, operator, etc.).
Administrators shall have the authority to manage user accounts, assign roles, and revoke access privileges.
User authentication shall be enforced to prevent unauthorized access to backup data and system settings.

Security-

Authentication:
Implement robust authentication mechanisms, including username/password combinations, biometric verification(future scope), or multi-factor authentication.
Ensure that only authorized users can access the system by verifying their identity through appropriate authentication methods.

Authorization:
Enforce role-based access control (RBAC) to restrict users' privileges based on their roles within the organization.
Define and assign permissions to different user roles to ensure that users can only perform actions relevant to their responsibilities.

Data Encryption:
Utilize encryption techniques to secure sensitive data both in transit and at rest.
Implement strong encryption algorithms to protect against unauthorized access or interception of confidential information.

Secure Communication:
Employ secure communication protocols (e.g., HTTPS, SSL/TLS) to encrypt data transmitted between clients and servers.
Ensure that data exchanged between different components of the system is encrypted to prevent eavesdropping or tampering.

Access Control:
Implement access controls at various levels of the system to restrict unauthorized access to sensitive functionalities or data.
Regularly review and update access control policies to adapt to changing security requirements and user roles.

Security Auditing and Monitoring:
Conduct regular security audits and vulnerability assessments to identify and remediate potential security weaknesses.
Implement logging mechanisms to record user activities, system events, and security-related incidents for auditing and forensic analysis purposes.

Performance-

Response Time:
Define acceptable response time benchmarks for critical operations such as account inquiries, transaction processing, and report generation.
Optimize system components and database queries to minimize response times and enhance user experience.

Throughput:
Ensure that the system can handle concurrent user requests and transactions efficiently, maintaining high throughput levels during peak usage periods.
Implement caching mechanisms, database optimizations, and asynchronous processing to improve throughput and reduce latency.

Concurrency Control:

Implement concurrency control mechanisms to manage simultaneous access to shared resources and prevent data inconsistencies or conflicts.
Utilize locking mechanisms, optimistic concurrency control, or transaction isolation levels to ensure data integrity and consistency during concurrent operations.

Maintainability-

Modularity:
Design the system with a modular architecture, separating functionality into cohesive modules or components.
Encapsulate modules with clear interfaces and well-defined responsibilities to facilitate independent development, testing, and maintenance.

Code Quality:
Enforce coding standards, best practices, and code review processes to ensure high code quality and readability.
Document codebase with comments, inline documentation, and README files to aid understanding and maintenance by developers.

Documentation:
Maintain comprehensive documentation, including system architecture diagrams, API documentation, user manuals, and troubleshooting guides.
Keep documentation up-to-date with changes to the system, APIs, configurations, and deployment procedures to facilitate ongoing maintenance and support.

Flexibility-

Customization:

Provide customizable features and configurations to adapt the system to the unique requirements and preferences of different banks or financial institutions.
Allow administrators to configure settings such as user roles, permissions, workflows, and reporting options to tailor the system to specific business needs.

Extensibility:
Design the system with extensible architectures and modular components that support future enhancements, additions, and customizations.
Provide well-defined extension points, hooks, or plugin frameworks to allow developers to extend system functionality through custom modules or integrations.
Adaptability:
Build the system with flexible architectures and design patterns that can accommodate evolving business requirements, regulatory changes, and technological advancements.
Implement design principles such as loose coupling, abstraction, and separation of concerns to enhance the system's adaptability and maintainability over time.
.

5. System Constraints

5.1 Hardware Constraints:

Server specifications:

Minimum: Quad-core CPU (2.5 GHz), 16 GB RAM, 250 GB SSD

Recommended: 8-core CPU (3.0 GHz), 32 GB RAM, 500 GB SSD

Network infrastructure:

- High-speed internet (1 Gbps minimum)
- Load balancing and failover
- Secure network segmentation

Supported devices:

- Desktops/laptops (modern browsers)
- Mobile (Android/iOS) with dedicated apps

5.2 Software Constraints:

Operating system: Linux (Ubuntu, Red Hat)

Database: High-performance, scalable DBMS (PostgreSQL, Oracle)

Middleware: Secure communication protocols (HTTPS, TLS) and API Gateway for external integrations.

5.3 Performance Constraints:

- **Response times:** Web: under 3 seconds, Mobile: under 5 seconds
- **Transaction throughput: 1,000 transactions per second (adjust based on transaction type, volume, and future growth)**
- **Scalability:** Horizontal scaling to accommodate increased load

5.4 Security Constraints:

- **Authentication/authorization:** Multi-factor authentication (MFA) and role-based access control (RBAC).
- **Encryption:** Industry-standard encryption (AES-256) for data at rest and in transit.
- **Compliance:** PCI DSS, GDPR, and any additional regional regulations.

5.5 Legal and Regulatory Constraints:

- Adherence to local/national banking regulations, data protection laws (e.g., GDPR, CCPA), and relevant financial industry standards.

5.6 Operational Constraints:

- **Availability:** 99.9% uptime for critical functions; planned maintenance outside peak hours.
- **Backup and recovery:** Regular backups to secure off-site locations and a disaster recovery plan for quick outage recovery.

5.7 Cultural and Organizational Constraints:

- **User training:** Materials on secure system usage and ongoing training for new features and updates.
- **Change management:** Well-defined processes for managing system changes and upgrades.
- **Accessibility:** System accessible to users with disabilities.

5.8 Environmental Constraints:

- **Data center:** Secure, reliable data center with appropriate environmental controls.
- **Disaster recovery:** Plan for recovering from natural disasters or other disruptions.
- **Environmental considerations:** System designed to minimize power consumption and heat generation.

5.9 Budgetary Constraints:

- Defined project budget to be adhered to.

- Cost-effective solutions for hardware, software, and other resources, while maintaining quality and security.

5.10 Interoperability Constraints:

- **Integration:** Ability to integrate with existing financial systems and payment networks.
- **Compliance:** Compatibility with relevant industry standards (ISO, SWIFT).