Software Requirements

Document

For

BorrowNest

Table of Contents

[**1. Introduction**](#_aa1vzfgdte90) **3**

[**2. Overall Description:**](#_lxfxvwcyryql) **3**

[**3. Requirements:**](#_sr25bl3hcb4z) **4**

[**4. Non-functional requirements:**](#_3rzepog2gsho) **6**

# Introduction

* 1. Purpose: The purpose of BorrowNest’s community rental marketplace is to allow for the rental of everyday household items and more, with the goals of reducing wastage, increasing the accessibility of items, and allowing for extra income for renters
  2. Scope: BorrowNest aims to provide a unified platform for renting out everyday household items, making it easier for people to access items they need temporarily without the burden of purchasing them. The platform will cater to both lenders, who can earn additional income by renting out items they own but do not frequently use, and renters, who can save money by borrowing items for short-term use.

# Overall Description:

* 1. Product features: The system will offer the following core functionalities:
* Item Listings: Lenders can post household items available for rent, with detailed descriptions, photos, rental rates (hourly, daily, weekly, etc.), and availability.
* Search and Filter: Renters can search for items by location, price, rental duration, and other filters to quickly find what they need.
* User Ratings and Reviews: A rating system for both lenders and renters, ensuring a trusted community where transactions are based on user feedback.
* Chat Functionality: Users will be able to communicate directly through the platform to clarify item details or arrange logistics.
* Flexible Rental Terms: Lenders can choose to offer items for local pickup, delivery, or both, with flexible pricing options.
* Moderation and Quality Control: BorrowNest will moderate listings to ensure items are in good working condition, maintaining high standards for customer satisfaction.
* Security Deposits and Insurance: For high-value items, a security deposit system and mandatory insurance will be in place to safeguard both lenders and renters.
  1. User classes: The system will have several user classes performing various roles. This is to ensure the permissions fencing so that each user can perform their role effectively. The base user will be able to both search for items to rent as well as offer their own items. Moderators and customer service representatives will have heightened roles to vet users/listings and assist with any problems that arise, respectively
  2. Constraints:

Legal and Ethical Constraints: BorrowNest must comply with local and international regulations regarding rental services, user privacy, and consumer protection. Data privacy laws (such as GDPR) will govern how personal and transaction data is collected, stored, and shared.

Security Requirements: Given that transactions and personal information will be exchanged on the platform, BorrowNest will implement secure data transmission, encryption for sensitive information, and login fail-safes. A system for handling disputes and security deposits will require careful management to avoid misuse.

Performance Overheads: Features such as real-time item availability and chat functionality may add processing overhead, requiring optimization to ensure smooth performance, especially under high user load.

Version Control and Maintenance: BorrowNest’s development will be tracked via GitHub for source control, ensuring efficient collaboration between development teams. The platform will support updates via service packs or hotfixes to ensure continuous improvement and bug fixes.

Payment System Integration: The inclusion of secure payment gateways must comply with financial regulations and add a layer of complexity in integrating third-party services.

# Requirements:

* 1. Item listings:
     1. Description: This is the most essential feature of all, and is the basis of how the user will interact with the app. Listing creations are initiated in the frontend client and persisted in the database.
     2. Functional Requirements:
        1. Users shall be able to create a listing, add a photo, description, and price and any other information they want
        2. They also shall be able to update or delete it whenever they want, except for when a rental is active on that item
        3. Renters shall be able to save any items they want to keep tabs on
  2. Searching and filtering:
     1. Description: This feature will make for ease of use, enabling users to find what they want quickly
     2. Functional Requirements:
        1. Users shall be able to enter any search term, triggering the UI to update to show results related to the search term
        2. Users shall be able to select from a menu of filter categories, refining their search query further. The filter menu will have the following filters:
           1. Location
           2. Price: (range slider bar with ability to set min and max price on it)
           3. Category of item
           4. Date posted
           5. Product rating
           6. Seller rating
  3. User ratings and reviews:
     1. Description: This feature will help renters build credibility, allowing users to make more informed decisions on the products they are about to rent
     2. Functional Requirements:
        1. Users shall be able to rate both the user renting out the product and the product itself
        2. Users shall be able to choose whether they are anonymous or not when posting the review / rating
        3. As product ratings are mandatory, they shall be enforced by not allowing further activity until the product review is submitted
  4. Chat functionality:
     1. Description: This feature will allow users to get to know more about the product being offered
     2. Functional Requirements:
        1. From a product listing page, users will be able to start a conversation with the owner. The conversation shall be specific to that product
        2. Users and owners alike shall get notifications for new messages
        3. Users shall have the ability to archive chats after the rental period has ended
  5. Flexible renting terms:
     1. Description: This feature will allow for flexibility in terms of the rental period and delivery method (pickup vs delivery)
     2. Functional Requirements:
        1. Users shall be able to select, if the product owner allows, between multiple rental periods with varied pricing
        2. Users shall be able to select, if the product owner allows, between the delivery method being pickup or delivery, with the delivery / return fees being borne by them in the latter case
  6. Moderation and quality control:
     1. Description: This feature will allow moderators to step in when necessary, especially during the listing create and customer service stages
     2. Functional Requirements:
        1. Moderators shall be able to mark incoming listing as ‘Needing Review’ if they deem it necessary to do so for listings that fail to comply with company policies
        2. They shall be able to take down accounts of users that are violating company policy
        3. They shall be able to make any changes on an account of a user requiring assistance, after gaining approval to do so
  7. Security deposits and insurance:
     1. Description: This feature will make for peace of mind for users, with renters getting assurance that they will have their product back and in the original condition after rental or get recompensed, or users against renters who never release their item after payment
     2. Functional Requirements:
        1. Users and renters alike shall be able to upload their insurance to their profile
        2. The system shall be able to hold payment until the rental has completed, upon which the payment will be released
        3. The system shall be able to take security deposits from the person requesting the rental

# Non-functional requirements:

* 1. Performance Requirements: The system should perform well under high load and have very little percentage error as people’s personal possessions are on the line. The mobile application, since it is to be supported on both iOS and Android, should perform more or less the same between the two platforms. Since there is a considerable amount of overhead involved with managing queues for each product, payment holds, and tracking delivery statuses, the server should have multi-core CPUs to handle concurrent tasks efficiently and must include sufficient RAM to support in-memory processing for managing queues and real-time data. It is essential that the server uses SSD storage to ensure fast read/write speeds, especially for database operations. The server must be scalable, both vertically for hardware upgrades and horizontally through load balancing, to accommodate future growth. The system should integrate robust queue management tools like RabbitMQ or Kafka, and must optimize database performance with connection pooling and clustering capabilities. High bandwidth and low latency are crucial for handling traffic, particularly for payment processing and real-time tracking. Strong security measures such as encryption, firewalls, and secure payment gateways must be implemented to safeguard sensitive transactions. The server must support high availability through redundancy and failover systems to ensure uptime, and should include real-time monitoring and centralized logging for performance tracking. A load balancer and API gateway should be used to distribute traffic and manage APIs effectively. Finally, it is recommended to assess whether a cloud platform or on-premise server best fits the scalability and compliance requirements of the project.
  2. Security Requirements: Security must be a top priority, especially with multiple microtransactions occurring simultaneously on the platform. The server must implement end-to-end encryption for all data exchanges, particularly during payment processing, to prevent unauthorized access. It is crucial to use secure payment gateways that comply with PCI-DSS standards to ensure safe handling of financial data. Authentication mechanisms like multi-factor authentication (MFA) should be enforced to protect user accounts. Additionally, the server should maintain secure storage for sensitive data with encrypted databases and backups. Real-time security monitoring must be employed to detect and mitigate potential threats, and firewalls and intrusion detection systems must be in place to safeguard against external attacks. Regular security audits and patch updates are essential to keep the system resilient against evolving threats.

**Functional Requirements Table**

| **Req. #** | **Input** | **Process** | **Output** |
| --- | --- | --- | --- |
| Item Listing | **Description, price, rental ranges offered, delivery method, photos and optional extra info if desired** | **Have a product creation form where the product owner enters all these details.** | **Listing preview as it appears to the user displayed after successful saving to the database** |
| **Searching and Filtering** | **Any parameters that the user has chosen to filter by** | **Apply selected filters to search query in backend** | **Return result of query (0 or more products) to the frontend for display** |
| **User ratings and reviews** | **Rating along with comment if desired** | **Take the rating and comment and relate it to the target product / user in the database.** | **Update the user / product profile in the frontend to reflect the addition of a new review. Have the first few review displayed be a mix of positive and negative reviews** |
| | **Chat functionality** | | --- |  |  | | --- | | **Message** | **Take message and send to target user. Trigger an action by the frontend listener** | **Frontend listener will pickup the new message and notify the target user** |
| | **Flexible rental terms** | | --- |  |  | | --- | | **Time period, price for time period, delivery method.** | **Take payload and create new product record in database. Return JSON representation of product to frontend.** | **Take JSON representation from backend and display it nicely** |
| **Moderation and quality control** | **Whatever is required to assist the customer** | **Dependent on specific customer service scenario** | **Dependent on specific customer service scenario** |
| **Security deposit and insurance** | **Insurance proof, amount of deposit, bank info on file** | **Take bank info and do a ACH withdrawal to charge the user** | **Display result of transaction result in transaction history** |

**Traceability Matrix**

| **Business Requirements Document (BRD)** | **Functional Requirements Document (FSD)** | **Priority** | **Test Case** | **Verification Method** | **Trace to Use Case** |
| --- | --- | --- | --- | --- | --- |
| **BR\_1 Product Creation Module** | **FR\_1 create products** | **High** | **TC001** | **System test, UI, functional test** | **UC\_001 (product CRUD)** |
|  | **FR\_2 update products** | **High** | **TC002** | **Functional test** | **UC\_001 (product CRUD)** |
|  | **FR\_3 view product details** | **High** | **TC003** | **UI and functional test** | **UC\_001 (product CRUD)** |
|  | **FR\_4 update status of product** | **Medium** | **TC004** | **Functional test, API test** | **UC\_001 (product CRUD)** |
|  | **FR\_5 favorite products** | **High** | **TC005** | **Functional and UI testing** | **UC\_001 (product CRUD)** |
| **BR\_2 Searching Module** | **FR\_6 search with filters** | **Medium** | **TC006** | **UI testing, functional test** | **UC\_002 (search and filter Product Listings)** |
|  | **FR\_7 search with keyword** | **High** | **TC007** | **UI testing, functional test** | **UC\_002 (search and filter Product Listings)** |
|  | **FR\_8 search with both** | **Medium** | **TC008** | **UI testing, functional test** | **UC\_002 (search and filter product listings)** |
| **BR\_3 Chat Module** | **FR\_9 renter to owner conversations** | **High** | **TC009** | **Functional test and UI test** | **UC\_003 (Chat functionality)** |
|  | **FR\_10 user to customer support conversations** | **Medium** | **TC010** | **UI and functional testing** | **UC\_003 (Chat functionality)** |
| **BR\_4 Rental Terms Module** | **FR\_11 rental terms specification on product** | **High** | **TC011** | **UI and functional** | **UC\_001 (Product CRUD)** |
|  | **FR\_12 ACH withdrawal from bank for delivery case** | **High** | **TC012** | **UI, functional, and security test** | **UC\_004 (delivery method handling)** |
|  | **FR\_13 delivery status updates** | **Medium** | **TC013** | **UI, functional, and security test** | **UC\_004 (delivery method handling)** |
| **#5: Profile creation module** | **FR\_14 create / update user profile** | **Medium** | **TC014** | **Functional and UI testing** | **UC\_005 (Profile CRUD)** |
| **#6: Owner/Renter protection Module** | **FR\_15 Insurance policy file upload for renter profile** | **Medium** | **TC015** | **Functional and UI testing** | **UC\_005 (Profile CRUD), UC\_006 (owner and renter protection)** |
|  | **FR\_16 Insurance policy file upload on product listing** | **Medium** | **TC016** | **Functional and UI testing** | **UC\_005 (Profile CRUD), UC\_006 (owner and renter protection)** |
|  | **FR\_17 ACH security deposit withdrawal** | **Medium** | **TC017** | **Functional and UI testing** | **UC\_006 (owner and renter protection)** |
|  | **FR\_18 Temporary Rental fee holding** | **Medium** | **TC018** | **Functional and UI testing** | **UC\_006 (owner and renter protection)** |

UML use case diagram:   
  
