Borrow Box – Final Report

FANTASTIC FOUR:

Fazal Zaman 22i-2362

Heer Fatima 22i-2371

Muhammad Fawaz 22i-2340

Allah tawaqqal

Submission Date : 29th April 2025

1. Project Introduction

Borrow Box is a peer-to-peer product rental platform designed to simplify and digitalize the rental process for users and owners. The system allows users to list, browse, rent, and review products. It facilitates seamless interaction between renters and owners while empowering administrators to manage the system effectively.

2. Functional and Non-Functional Requirements

Functional Requirements

2.1 User Authentication & Management

- Users must be able to register using a valid email and password.
- Users can update their profile details and password.
- Admins can manage and deactivate user accounts.

2.2 Product Listing & Management

- Owners must be able to add, edit, and remove their rental items.
- Each product must have a title, description, price, availability status, and category.
- Users can report incorrect product details.

2.3 Rental Request & Approval

- Users can request to rent a product by specifying the rental duration.
- Owners can approve or reject rental requests.
- Approved rentals are marked as "Active" until completion.
- Renters can cancel a request before approval.

2.4 Report Generation & Analytics

- Admins should be able to generate detailed rental reports.
- Reports should include rental history, payment transactions, and user activity logs.
- Users should be able to download their rental invoices.
- Reports should support filtering by date range, user, and product category.
- The system should allow exporting reports in PDF, CSV, and Excel formats.

2.5 Feedback & Reviews

- Renters can leave feedback on completed rentals.
- Owners can respond to user reviews.
- Reviews should have a rating system (1-5 stars).

2.6 Search & Filtering

- Users should be able to search for rental items by name, category, price range, and availability.
- Advanced filters should be available for sorting by highest-rated, lowest price, etc.

2.7 Rental History Tracking

- Users should have access to their rental history, including past payments and completed transactions.
- Admins should be able to generate rental reports.

2.8 Admin Dashboard & System Management

- Admins should be able to monitor active rentals and user statistics.
- System logs should be maintained for auditing purposes.

2.9 Dispute Resolution

- Users should be able to report rental issues (e.g., damaged product, fraudulent listings).
- Admins should be able to review and resolve disputes.

Non-Functional Requirements

3.1 Product Requirements

- The system should load pages within **2 seconds** on a standard internet connection.
- It must maintain an **uptime of 99.9%**.
- All transactions must be encrypted using **SSL/TLS**.
- The system should have a **modular architecture** to support easy updates and maintenance.
- **Scalability**: The system should efficiently handle a growing number of users and transactions.
- **Error Handling**: Clear error messages should be displayed for invalid actions.
- **Logging & Monitoring**: The system should maintain logs for debugging and security audits.

3.2 Organizational Requirements

- The system must follow **Agile development methodologies**.
- It should be built using **NEXT.js** for **backend** and **frontend**.
- The database should be optimized for **high-volume queries**.
- The system should support **multi-language localization** to cater to a global user base.
- It should provide **role-based access control (RBAC)** to restrict access to different system functionalities.

3.3 External Requirements

- **Security Compliance**: The system must comply with **GDPR** for user data protection.
- **Cross-Browser Compatibility**: The UI should work on all major browsers.
- **Third-Party API Support**: The system should allow integration with third-party services for extended functionality.
- **Backup & Disaster Recovery**: Data backups should be taken at regular intervals to prevent data loss.

3. User Stories

User Story 1: Register

Preconditions:

- The user does not have an existing account.
- The system is online and operational.

Postconditions:

- A new user account is created.
- The user receives a confirmation email.
- The user can log in to the system.

User Story 2: Browse Product

Preconditions:

- The user is logged in.
- There are products available for rent.

Postconditions:

- The user can view the list of products.
- The system displays product details based on the user's search or filters.

User Story 3: Request Rental

Preconditions:

- The user is logged in.
- The requested product is available for rent.

Postconditions:

- The rental request is sent to the product owner.
- The user receives a confirmation or rejection notification.

User Story 4: View Rentals History

Preconditions:

- The user is logged in.
- The user has rented products in the past.

Postconditions:

- The system displays the user's rental history.

User Story 5: Report Issue

Preconditions:

- The user is logged in.
- The user has an issue with a rental or transaction.

Postconditions:

- The issue report is submitted.
- The admin or owner receives the report.
- The user is notified about the status of the issue.

User Story 6: Give Feedback for Product

Preconditions:

- The user is logged in.
- The user has completed a rental for the product.

Postconditions:

- The feedback is stored in the system.
- Other users can view the feedback on the product page.

User Story 7: List Product for Rent

Preconditions:

- The owner is logged in.
- The owner has a product they want to rent out.

Postconditions:

- The product is listed in the system.
- Renters can see and request the product.

User Story 8: Manage Products

Preconditions:

- The owner is logged in.
- The owner has at least one listed product.

Postconditions:

- The product details are updated or deleted based on the owner's action.

User Story 9: Resolve Issues

Preconditions:

- The admin/owner is logged in.
- There is at least one unresolved issue reported by a user.

Postconditions:

- The issue is marked as resolved or updated with status changes.
- The affected user is notified.

User Story 10: Manage Users

Preconditions:

- The admin is logged in.
- There are registered users in the system.

Postconditions:

- The admin can view, block, or deactivate user accounts.
- The affected user's account status is updated.

User Story 11: Generate Reports

Preconditions:

- The admin is logged in.
- There is relevant data available for reporting.

Postconditions:

- The system generates and displays the requested report.

User Story 12: Admin Management

Preconditions:

- The admin is logged in.
- The system is functioning correctly.

Postconditions:

- The admin has performed actions like monitoring, managing settings, or handling issues.

4. Product Backlog

Product Backlog

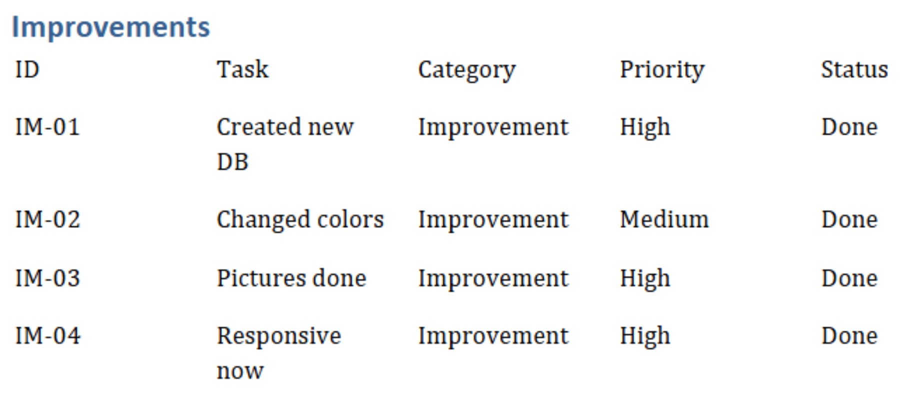
User Stories

ID	As a...	I want to...	So that...	Priority	Status
US-01	New user	Register with email and password	Access the platform	High	Done
US-02	Registered user	Log in securely	Access my account	High	Done
US-03	Renter	Search and filter rental products	Find what I need	High	Done
US-04	Product owner	List items for rent	Renters can request them	High	Done
US-05	Renter	Send a rental request	Use product for a specified time	High	Done
US-06	User	See my past rentals	Track my transactions	Medium	Done

US-07	Renter	Leave feedback and ratings	Help future users make informed decisions	Medium	Done
US-08	Product owner	Edit or delete my rental listings	Keep inventory updated	Medium	Done
US-09	Product owner	Edit or delete my rental listings	Keep inventory updated	Medium	Done
US-10	Admin	Generate reports on transactions and users	Analyze platform performance	Low	Done
US-11	Renter	Raise disputes about a rental issue	Get support from the platform	Medium	Done

Bugs and Issues

ID	Task	Category	Priority	Status
BI-01	Correct the DB	Bug	High	Done
BI-02	Change the color scheme of UI	Bug	Medium	Done
BI-03	Pictures are not loading	Bug	High	Done
BI-04	Make the web responsive	Bug	High	Done



Improvements

ID	Task	Category	Priority	Status
IM-01	Created new DB	Improvement	High	Done
IM-02	Changed colors	Improvement	Medium	Done
IM-03	Pictures done	Improvement	High	Done
IM-04	Responsive now	Improvement	High	Done

Refer to link below to Trello board for full product backlog.

Product_Backlog.docx

5. Sprint Backlog

Sprint Backlog

Sprint 1

To Do:

- User login
- User Registration
- Browse Product
- List Product for Rent
- Create Database
- Documentation

Doing:

- User login
- User Registration
- Browse Product
- List Products
- Working with Database
- Documentation

Done:

- User login
- User Registration
- Browse Product
- List Product for Rent
- Database (Incomplete)
- Documentation Done

Sprint 2

To Do:

- Request Rental
- View Rental History
- Cart Issue
- Report Issue
- Give Feedback
- Create New Database
- SRS Documentation
- Keep GitHub Updated

Doing:

- Request Rental
- View Rental History
- Cart Issues
- Report Issues
- Give Feedback
- New Database Done
- SRS Documentation
- Made a New Push to GitHub

Done:

- Request Rentals
- View Rental History
- Cart Issue
- Report Issue
- Give Feedback
- Database Ready
- Done with SRS Documentation
- GitHub Updated

Sprint 3

To Do:

- Implement Admin Controls
- Manage Products
- Manage Users
- Generate Reports
- Software Project Plan
- Architecture Diagrams
- Black Box Testing
- White Box Testing
- Final Report

Doing:

- Implement Admin Controls
- Manage Products
- Manage Users
- Generate Reports
- Software Project Plan
- Architecture Diagrams
- Black Box Testing

- White Box Testing
- Final Report

Done:

- Implement Admin Controls
- Manage Products
- Manage Users
- Generate Reports
- Software Project Plan
- Architecture Diagrams
- Black Box Testing
- White Box Testing
- Final Report

Refer to above link for complete Sprint Backlog

sprint_backlog.docx

6. Project Plan

Software Project Plan Project:

Borrow Box - A Peer-to-Peer Product Rental Platform

Team Members:

22i-2362 Fazal Zaman

22i-2371 Heer

22i-2340 M. Fawaz

Submission Date: April 28, 2025

1. Work Breakdown Structure (WBS)

Level 1: Borrow Box Development Lifecycle

1.0 Requirements Phase

1.1 Requirements Gathering

1.2 SRS Documentation

1.3 Functional & Non-Functional Requirement Analysis

1.4 Product Backlog Creation

1.5 Trello Board Initialization and Task Assignment

2.0 Design Phase

2.1 UI/UX Design - Login, Register, Navigation, Footer

2.2 Wireframes for product views and interactions

2.3 UML Diagrams - Use Case Diagram - Sequence Diagrams - Class Diagram - Package Diagram

2.4 System Architecture and Deployment Planning

3.0 Sprint 1: Core Functional Development

3.1 User Registration and Login

3.2 User Profile Management

3.3 Session Handling and Auth Guard

3.4 Sprint 1 Backlog Implementation and Review

4.0 Sprint 2: Product and Dashboard Modules

4.1 Product Listing and CRUD (Create, Read, Update, Delete)

4.2 Product Search and Filtering

4.3 Admin Dashboard Software Project Plan - Borrow Box

4.4 Sprint 2 Backlog Implementation and Review

5.0 Sprint 3: Testing and Finalization

5.1 Black Box Testing - Equivalence Partitioning and Boundary Value

5.2 White Box Testing - Code Coverage and Unit Analysis

5.3 Final Bug Fixes and UX Enhancements

5.4 Final Deployment and Feature Lock GANTT CHART

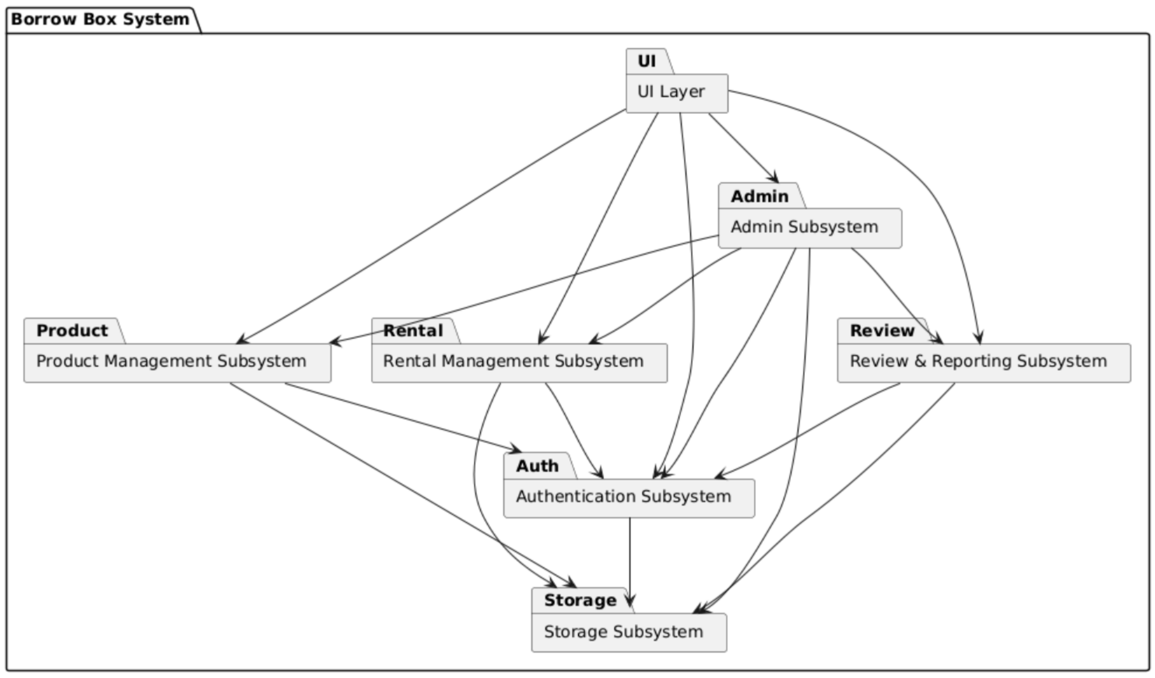
7. Architecture Diagram

System Architecture - Borrow Box

CS3009 - Software Engineering - Spring 2025

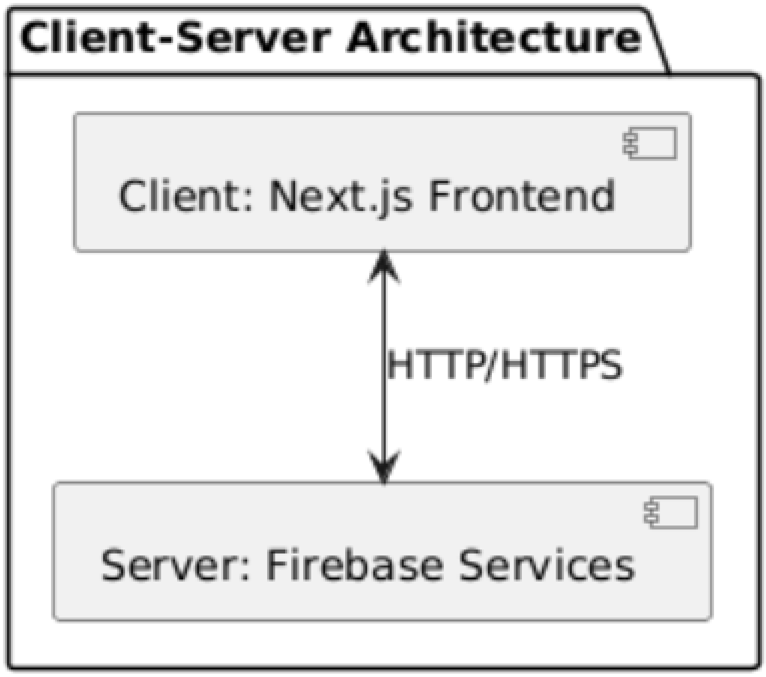
System Architecture : Borrow Box - A Peer-to-Peer Product Rental Platform

A. **Subsystem Identification**



B. Architecture Styles

The system uses Client-Server Architecture:

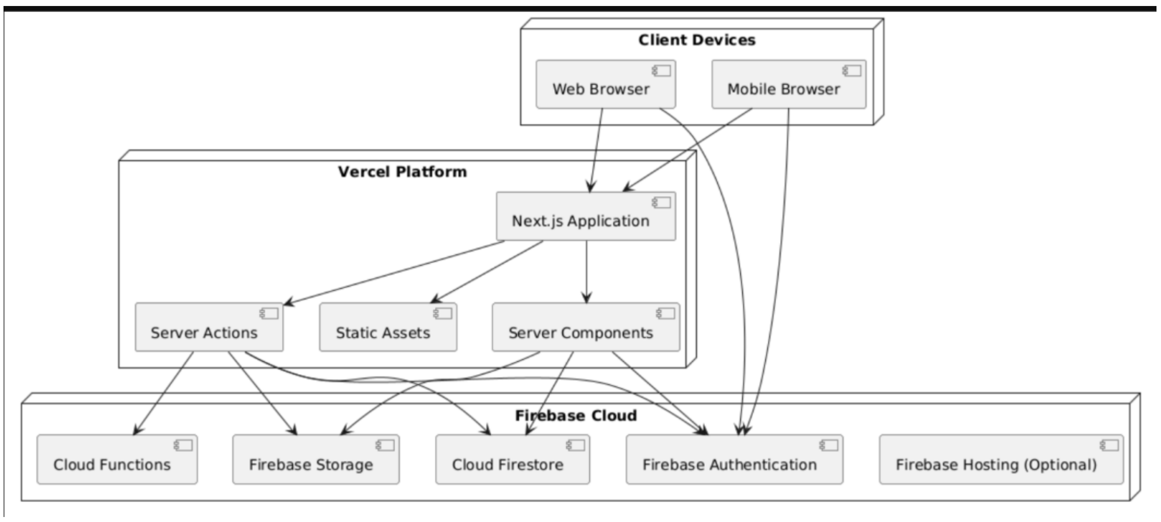


Frontend: Built using React components (e.g., Navbar, Footer, Hero) and communicates with the backend via REST APIs.

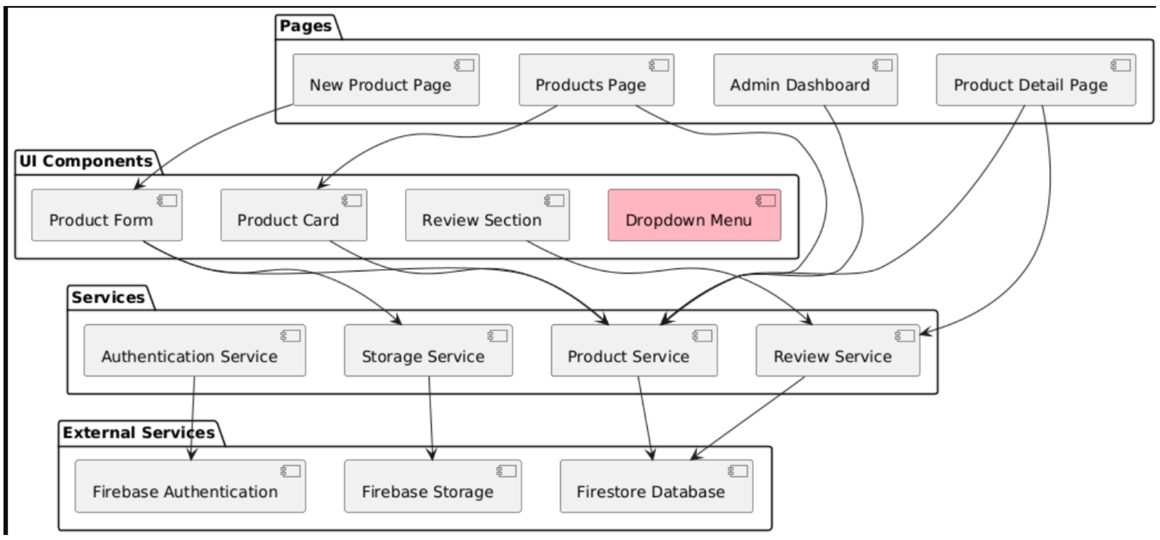
Backend: Implements a layered architecture with services like AuthService, ProductService, and RentalService.

Database: Relational database for storing users, products, and rentals.

C. Deployment diagram for client deployments



D. Component diagrams showing which services/components/subsystems you expect to be interfacing with or modifying in order to implement your enhancements.



8. Design (Sprint 3 Items)

The following design elements correspond to the tasks undertaken during Sprint 3:

1. Admin Controls

- **Design a secure and intuitive admin dashboard** to allow administrators to perform CRUD operations (Create, Read, Update, Delete) on products and users.
- **Role-based access control (RBAC)** will be implemented to ensure only authorized users access admin features.

2. Manage Products

- **Product Management Interface** includes:
  - Add new products with details (title, description, image, price).
  - Edit and delete existing products.
  - Search, filter, and sort product lists.
- **Backend Design:** APIs to handle product management operations securely.

3. Manage Users

- **User Management Panel** enables:
  - Viewing all registered users.
  - Editing user roles (e.g., upgrade to admin, disable accounts).
  - Deleting users if necessary.
- **Database Schema** will store user roles, status, and history of activities.

4. Generate Reports

- **Report Generation Module** designed to:
  - Generate reports on user activities, products, and system usage.
  - Export reports in formats like PDF/CSV.
- **Dynamic Filters** will allow admins to generate custom reports based on date ranges or categories.

5. Software Project Plan

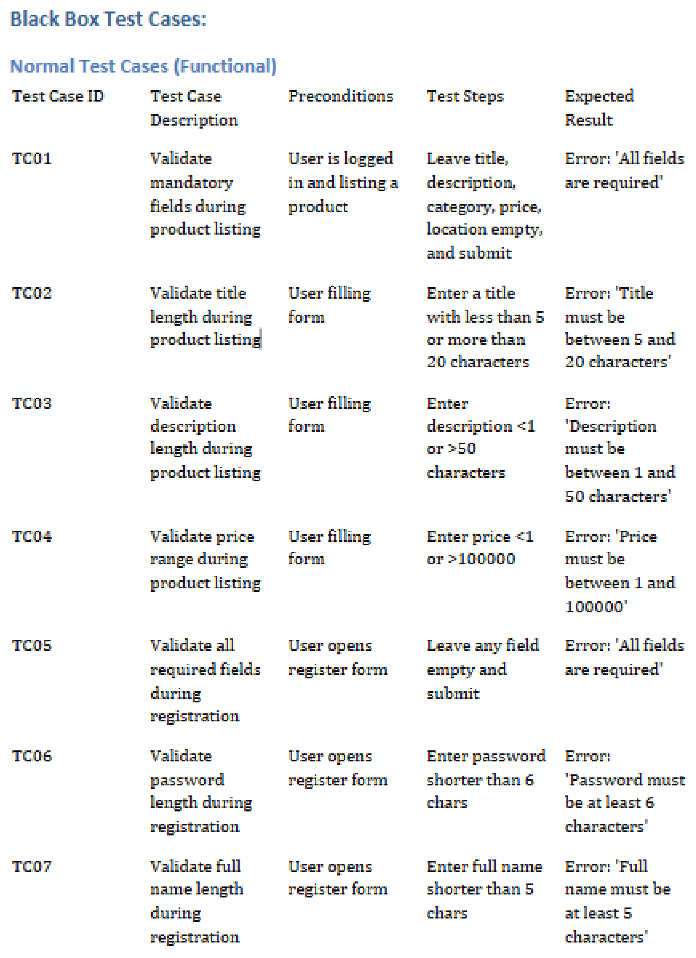
- **Planning Document** updates:

    - Reflect new features added during Sprint 3.
    - Update Gantt charts, milestones, risks, and mitigation strategies.

6. Architecture Diagrams

- **Update System Architecture**:
    - Include new modules (Admin Controls, Report Generator).
    - Draw updated component and deployment diagrams showing interactions.

7. Black Box Testing



Black Box Test Cases:

Normal Test Cases (Functional)

Test Case ID	Test Case Description	Preconditions	Test Steps	Expected Result
TC01	Validate mandatory fields during product listing	User is logged in and listing a product	Leave title, description, category, price, location empty, and submit	Error: 'All fields are required'
TC02	Validate title length during product listing	User filling form	Enter a title with less than 5 or more than 20 characters	Error: 'Title must be between 5 and 20 characters'
TC03	Validate description length during product listing	User filling form	Enter description <1 or >50 characters	Error: 'Description must be between 1 and 50 characters'
TC04	Validate price range during product listing	User filling form	Enter price <1 or >100000	Error: 'Price must be between 1 and 100000'
TC05	Validate all required fields during registration	User opens register form	Leave any field empty and submit	Error: 'All fields are required'
TC06	Validate password length during registration	User opens register form	Enter password shorter than 6 chars	Error: 'Password must be at least 6 characters'
TC07	Validate full name length during registration	User opens register form	Enter full name shorter than 5 chars	Error: 'Full name must be at least 5 characters'

TC08	Password prompt on product deletion	User tries to delete product	Delete product	Prompt user for password
TC09	Password + reCAPTCHA for admin creation	Admin promotes user	Try to make someone admin	Prompt for password and reCAPTCHA
TC10	Password + reCAPTCHA for user removal	Admin removes user	Try deleting a user	Prompt for password and reCAPTCHA
TC11	Verify successful product listing	Valid product info entered	Submit form	Product is listed successfully
TC12	Verify successful registration	Valid details entered	Submit register form	User is registered successfully

Boundary Value Test Cases

Test Case ID	Test Case Description	Preconditions	Test Steps	Expected Result
B01	Title exactly 5 characters (lower boundary)	User filling form	Enter 5-character title	Product accepted
B02	Title exactly 20 characters (upper boundary)	User filling form	Enter 20-character title	Product accepted
B03	Title 4 characters (below lower boundary)	User filling form	Enter 4-character title	Error: 'Title must be between 5 and 20 characters'
B04	Title 21 characters (above upper boundary)	User filling form	Enter 21-character title	Error: 'Title must be between 5 and 20 characters'

B05	Description 1 character (lower boundary)	User filling form	Enter 1-character description	Product accepted
B06	Description 50 characters (upper boundary)	User filling form	Enter 50-character description	Product accepted
B07	Description 0 characters (below boundary)	User filling form	Leave description blank	Error: 'Description must be between 1 and 50 characters'
B08	Description 51 characters (above boundary)	User filling form	Enter 51-character description	Error: 'Description must be between 1 and 50 characters'
B09	Price exactly 1 (lower boundary)	User filling form	Enter price = 1	Product accepted
B10	Price exactly 100000 (upper boundary)	User filling form	Enter price = 100000	Product accepted
B11	Price = 0 (below lower boundary)	User filling form	Enter price = 0	Error: 'Price must be between 1 and 100000'
B12	Price = 100001 (above upper boundary)	User filling form	Enter price = 100001	Error: 'Price must be between 1 and 100000'

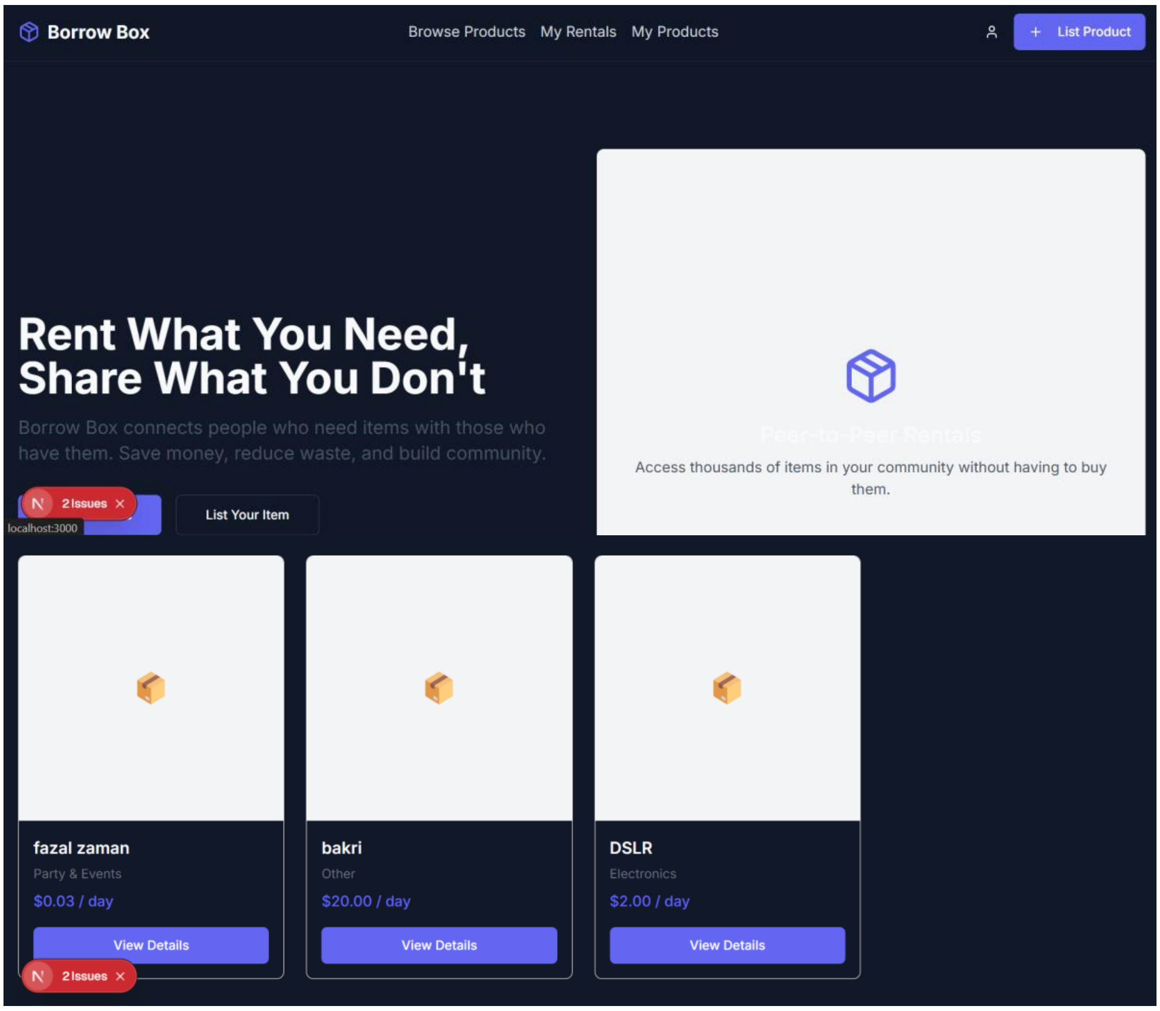
Negative Test Cases

Test Case ID	Test Case Description	Preconditions	Test Steps	Expected Result
N01	Submit empty product form	User on product form	Submit without filling fields	Error: 'All fields are required'
N02	Submit empty register form	User on register page	Submit without any input	Error: 'All fields are required'
N03	Passwords do not match on confirmation	User registering	Enter mismatching password and confirm password	Error: 'Passwords do not match'
N04	SQL Injection attempt in title field	User filling product form	Enter title: `'; DROP TABLE products; --`	Input rejected/sanitized
N05	Special characters in price field	User filling product form	Enter price = `abc$%#`	Error: 'Invalid price format'
N06	Exceed character limit in all fields	User filling form	Fill > allowed characters in title/description	Error: 'Maximum characters exceeded'
N07	Bypass reCAPTCHA during admin creation	Admin promoting user	Skip captcha step (tampering)	Action blocked, captcha required

8. White Box Testing

- **Write Unit Tests and Integration Tests**:
    - Focus on functions handling admin tasks.
    - Coverage for backend APIs, authentication, and authorization logic

9. Actual Implementation Screenshots



Rent What You Need, Share What You Don't

Borrow Box connects people who need items with those who have them. Save money, reduce waste, and build community.

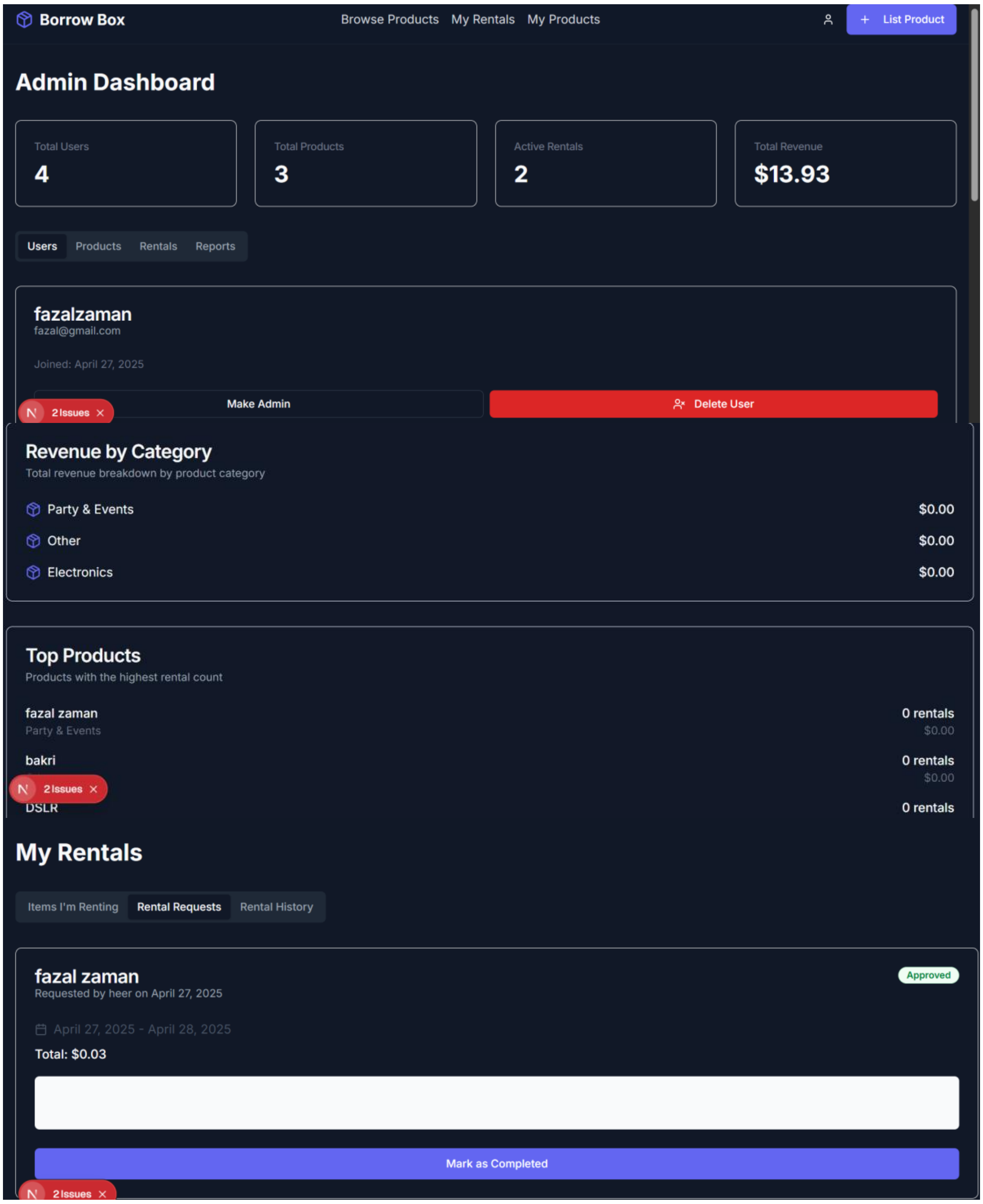
Peer-to-Peer Rentals

Access thousands of items in your community without having to buy them.

fazal zaman
Party & Events
$0.03 / day

bakri
Other
$20.00 / day

DSLR
Electronics
$2.00 / day


Borrow Box

Browse Products My Rentals My Products

List Product

Admin Dashboard

Total Users	Total Products	Active Rentals	Total Revenue
4	3	2	$13.93

Users Products Rentals Reports

fazalzaman
fazal@gmail.com

Joined: April 27, 2025

Make Admin | Delete User

2 Issues

Revenue by Category
Total revenue breakdown by product category

Party & Events	$0.00
Other	$0.00
Electronics	$0.00

Top Products
Products with the highest rental count

fazal zaman Party & Events	0 rentals $0.00
bakri	0 rentals $0.00
DSLR	0 rentals

2 Issues

My Rentals

Items I'm Renting | Rental Requests | Rental History

fazal zaman
Requested by heer on April 27, 2025

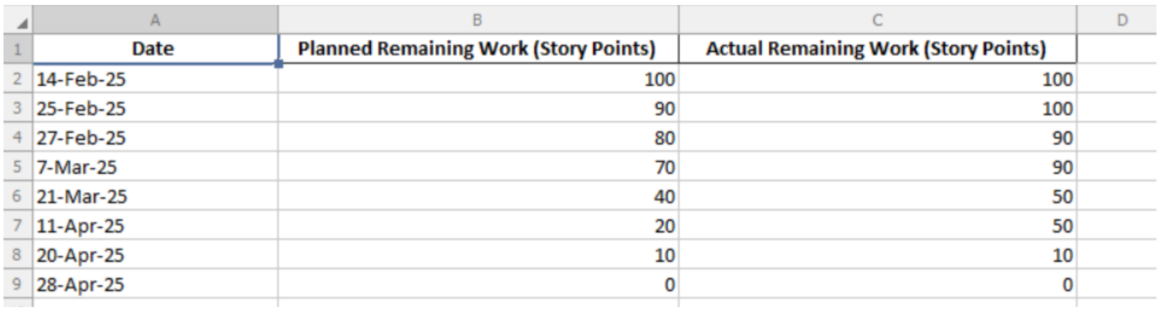
Approved

April 27, 2025 - April 28, 2025
Total: $0.03

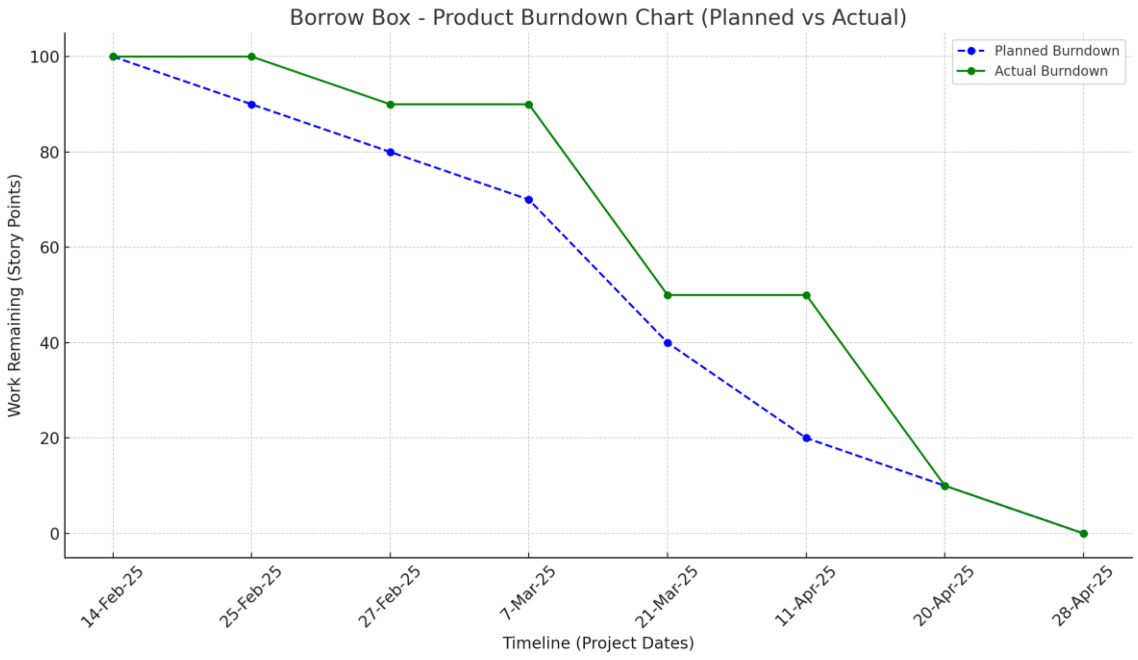
Mark as Completed

2 Issues

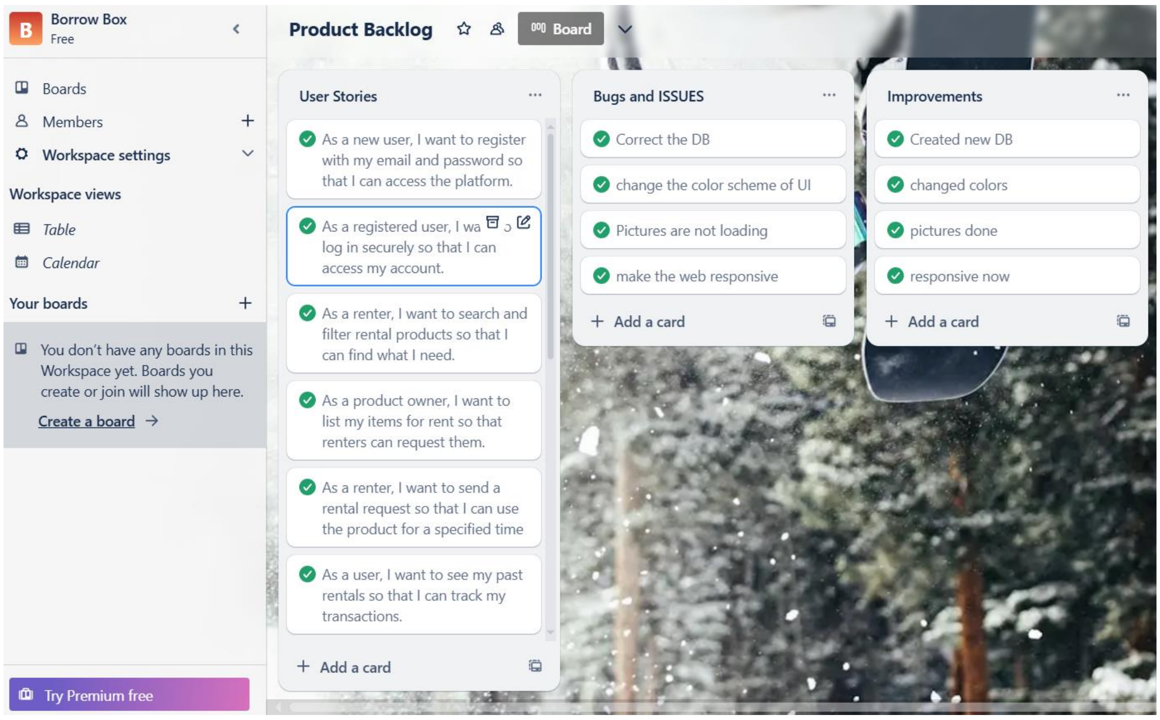
10. Product Burndown Chart



Date	Planned Remaining Work (Story Points)	Actual Remaining Work (Story Points)
14-Feb-25	100	100
25-Feb-25	90	100
27-Feb-25	80	90
7-Mar-25	70	90
21-Mar-25	40	50
11-Apr-25	20	50
20-Apr-25	10	10
28-Apr-25	0	0



11. Trello Board Screenshots



You don't have any boards in this Workspace yet. Boards you create or join will show up here.

User Stories

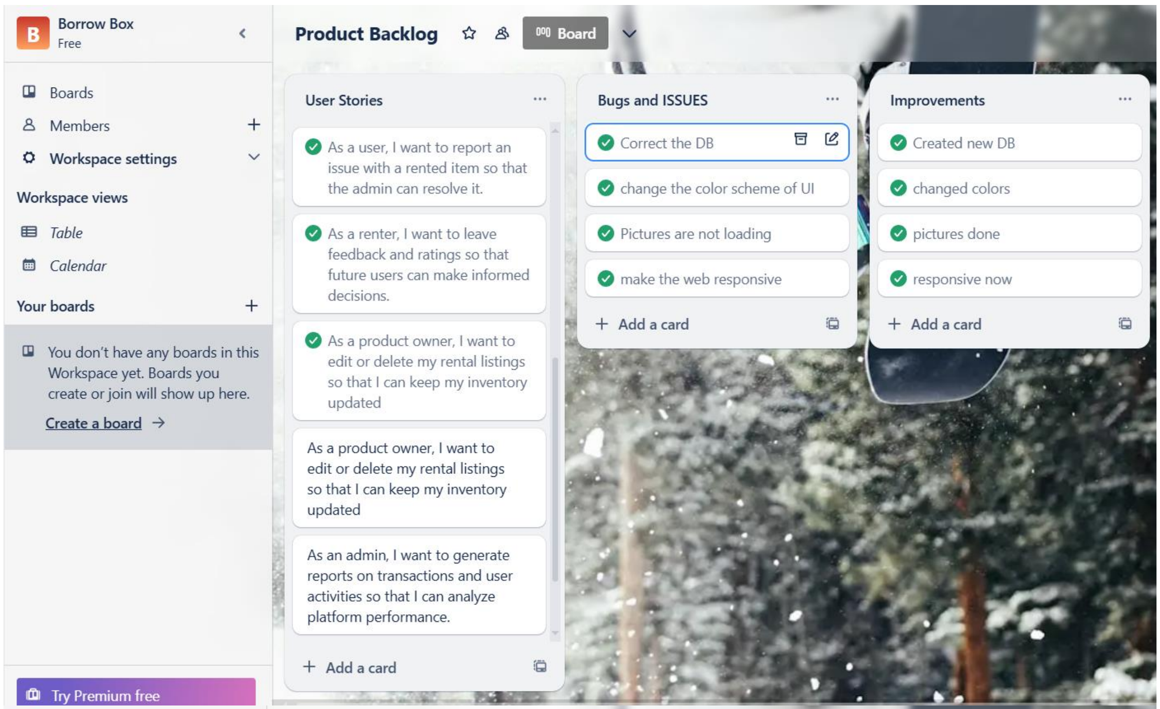
- ✓ As a new user, I want to register with my email and password so that I can access the platform.
- ✓ As a registered user, I wa[illegible] log in securely so that I can access my account.
- ✓ As a renter, I want to search and filter rental products so that I can find what I need.
- ✓ As a product owner, I want to list my items for rent so that renters can request them.
- ✓ As a renter, I want to send a rental request so that I can use the product for a specified time
- ✓ As a user, I want to see my past rentals so that I can track my transactions.

Bugs and ISSUES

- ✓ Correct the DB
- ✓ change the color scheme of UI
- ✓ Pictures are not loading
- ✓ make the web responsive

Improvements

- ✓ Created new DB
- ✓ changed colors
- ✓ pictures done
- ✓ responsive now



B Borrow Box
Free

Boards
Members
Workspace settings

Workspace views
Table
Calendar

Your boards

You don't have any boards in this Workspace yet. Boards you create or join will show up here.
Create a board →

Try Premium free

Product Backlog
Board

User Stories
- ✓ As a user, I want to report an issue with a rented item so that the admin can resolve it.
- ✓ As a renter, I want to leave feedback and ratings so that future users can make informed decisions.
- ✓ As a product owner, I want to edit or delete my rental listings so that I can keep my inventory updated
- As a product owner, I want to edit or delete my rental listings so that I can keep my inventory updated
- As an admin, I want to generate reports on transactions and user activities so that I can analyze platform performance.

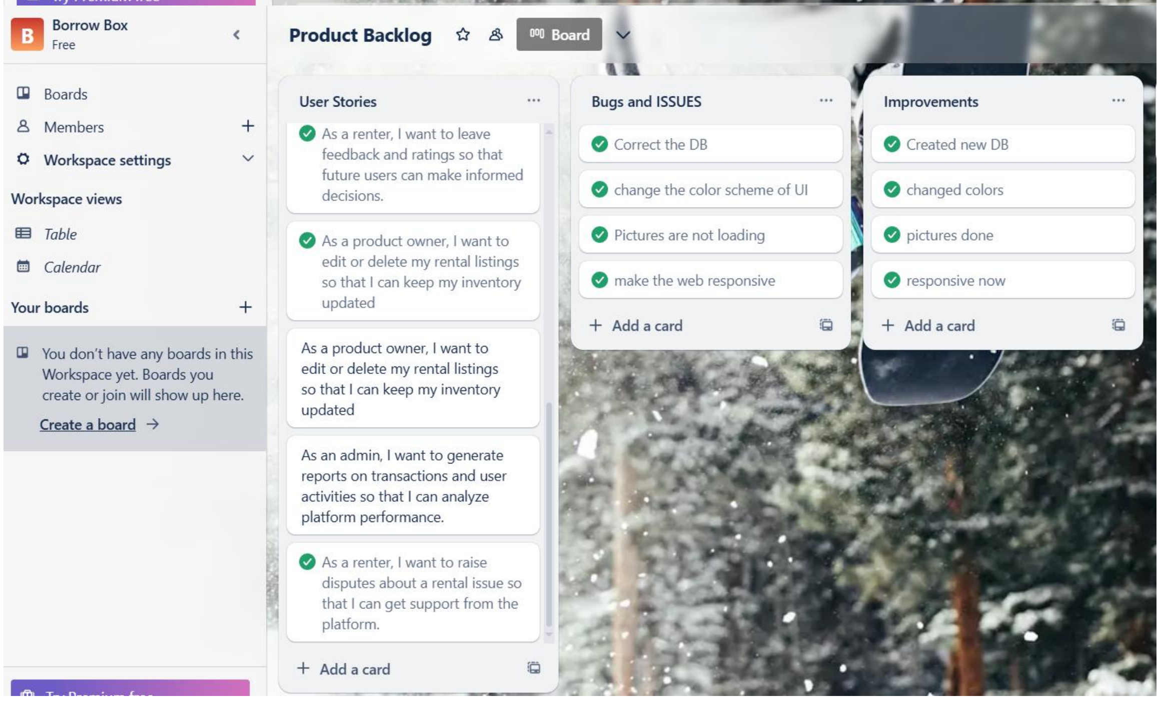
+ Add a card

Bugs and ISSUES
- ✓ Correct the DB
- ✓ change the color scheme of UI
- ✓ Pictures are not loading
- ✓ make the web responsive

+ Add a card

Improvements
- ✓ Created new DB
- ✓ changed colors
- ✓ pictures done
- ✓ responsive now

+ Add a card



B Borrow Box
Free

Boards
Members
Workspace settings

Workspace views
Table
Calendar

Your boards

You don't have any boards in this Workspace yet. Boards you create or join will show up here.
Create a board →

Product Backlog
Board

User Stories
- ✓ As a renter, I want to leave feedback and ratings so that future users can make informed decisions.
- ✓ As a product owner, I want to edit or delete my rental listings so that I can keep my inventory updated
- As a product owner, I want to edit or delete my rental listings so that I can keep my inventory updated
- As an admin, I want to generate reports on transactions and user activities so that I can analyze platform performance.
- ✓ As a renter, I want to raise disputes about a rental issue so that I can get support from the platform.

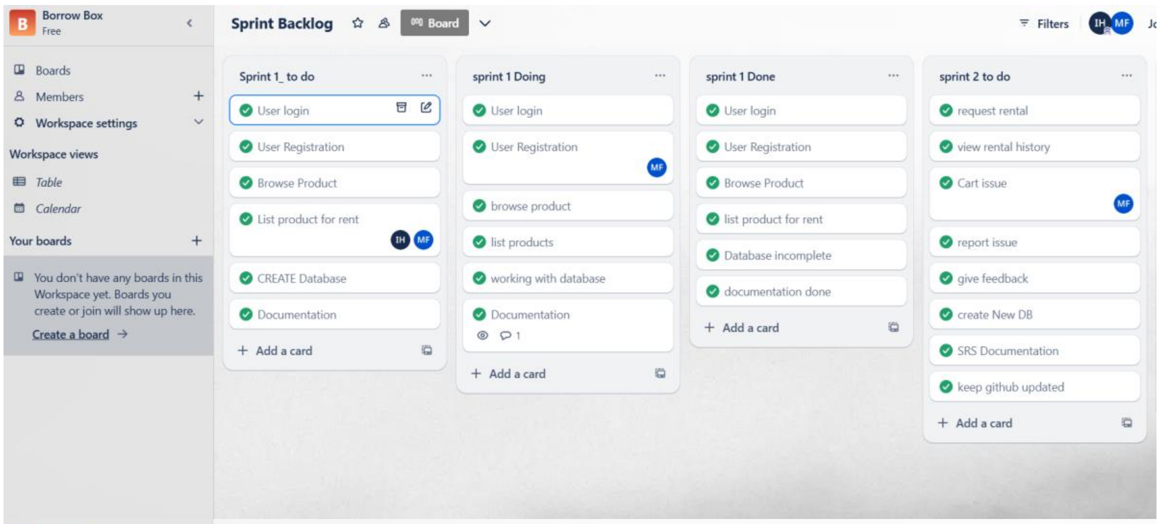
+ Add a card

Bugs and ISSUES
- ✓ Correct the DB
- ✓ change the color scheme of UI
- ✓ Pictures are not loading
- ✓ make the web responsive

+ Add a card

Improvements
- ✓ Created new DB
- ✓ changed colors
- ✓ pictures done
- ✓ responsive now

+ Add a card

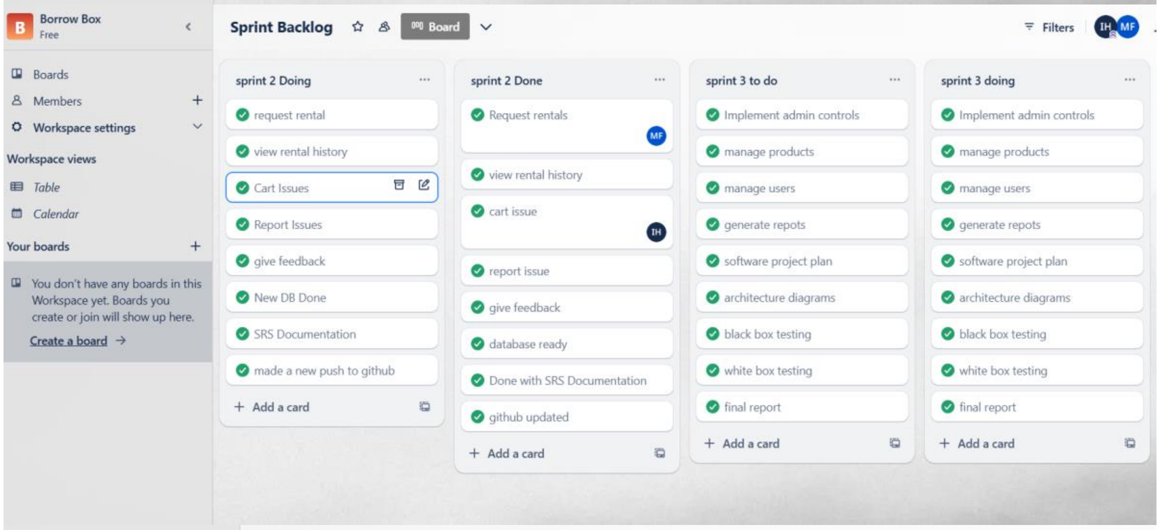


Sprint 1_ to do
- User login
- User Registration
- Browse Product
- List product for rent
- CREATE Database
- Documentation

sprint 1 Doing
- User login
- User Registration
- browse product
- list products
- working with database
- Documentation

sprint 1 Done
- User login
- User Registration
- Browse Product
- list product for rent
- Database incomplete
- documentation done

sprint 2 to do
- request rental
- view rental history
- Cart issue
- report issue
- give feedback
- create New DB
- SRS Documentation
- keep github updated

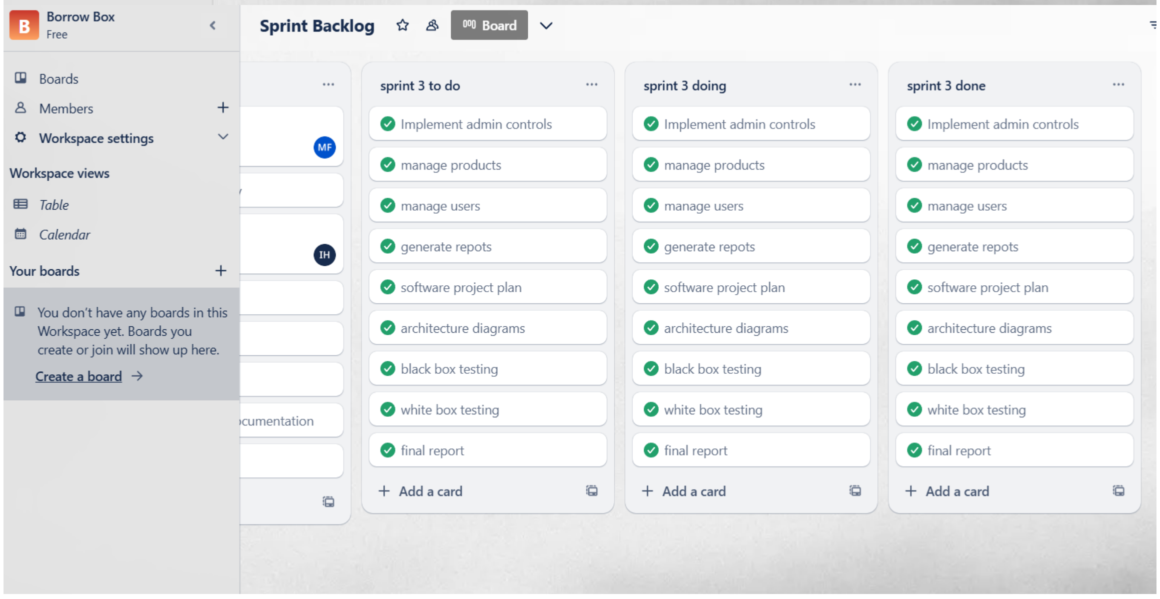


sprint 2 Doing
- request rental
- view rental history
- Cart Issues
- Report Issues
- give feedback
- New DB Done
- SRS Documentation
- made a new push to github

sprint 2 Done
- Request rentals
- view rental history
- cart issue
- report issue
- give feedback
- database ready
- Done with SRS Documentation
- github updated

sprint 3 to do
- Implement admin controls
- manage products
- manage users
- generate repots
- software project plan
- architecture diagrams
- black box testing
- white box testing
- final report

sprint 3 doing
- Implement admin controls
- manage products
- manage users
- generate repots
- software project plan
- architecture diagrams
- black box testing
- white box testing
- final report



sprint 3 to do
- Implement admin controls
- manage products
- manage users
- generate repots
- software project plan
- architecture diagrams
- black box testing
- white box testing
- final report

sprint 3 doing
- Implement admin controls
- manage products
- manage users
- generate repots
- software project plan
- architecture diagrams
- black box testing
- white box testing
- final report

sprint 3 done
- Implement admin controls
- manage products
- manage users
- generate repots
- software project plan
- architecture diagrams
- black box testing
- white box testing
- final report

https://trello.com/w/userworkspace69812780/home

12. Test Cases – Black Box

Black box tests based on user input validation, product flow, and rental system.

13. Test Cases – White Box

Unit tests with internal logic and function path coverage.

14. Work Division Between Group Members

Heer (22i-2371)

Role: Frontend, UI/UX Developer

Designed and implemented the Registration and Login Pages.

Developed the Search and Filter Functionality.

Implemented Feedback and Review Sections.

Fawaz (22i-2340)

Role: Backend Developer (Product)

Implemented Product CRUD Operations (Add, Edit, Delete Products).

Developed Rental Request and Approval Logic.

Integrated Backend APIs with Frontend.

Fazal (22i-2362)

Role: Admin Panel & Testing Lead

Built the Admin Dashboard for User and Dispute Management.

Handled Dispute Resolution Features.

Led Black Box and White Box Testing Activities

15. Lessons Learned

During the development of the Borrow Box project, the team collectively enhanced its technical skills, project management practices, and understanding of agile workflows. Working in collaboration helped improve communication, task division, and sprint planning abilities. The experience also deepened the team's knowledge of system architecture, testing techniques, and deployment processes.

Individual lessons learned:

Heer (22i-2371)

Improved at handling state management and component interactions in NEXT.js.

Gained a deeper understanding of frontend optimization and user experience design.

Fawaz (22i-2340)

Gained experience in designing modular, scalable APIs with Next.js.

Enhanced backend integration skills and database interaction handling.

Fazal (22i-2362)

Understood the complete process of Black Box and White Box Testing methodologies.

Developed expertise in generating unit tests, integration tests, and analyzing coverage reports.