1. Introduction & System Intent

Client Overview

Client Name: Papela

Address: Latter Day Saints Church, Hannah School Road, Accra

Contact: Yaa Nimfa-Osei Buadu (Manager) | buadunyo@gmail.com| +233 54 859 1362

Papela is a privately owned company that specializes in providing event equipment for various functions, including weddings, parties, and corporate events. The company offers a broad selection of rental items such as chairs, tables, canopies, napkins, table covers, and other event-related accessories. It serves both individual clients and institutions primarily within Accra and its surrounding areas.

Problem Statement

Currently, Papela manages bookings and inventory manually using physical notebooks, mobile calls, and messaging apps like WhatsApp. This outdated process often results in duplicated reservations, missing orders, and inefficient stock tracking. Delays in communication and limited visibility into item availability have led to poor customer experience and reduced operational efficiency, especially during peak event periods.

System Vision

A modern web-based rental management system that enables users to easily check item availability, place bookings, and manage rental operations through a centralized platform.

User Personas

Primary Users:

• Rental Staff – Responsible for handling bookings, tracking inventory levels, and logging item returns.

• Customers – Individuals or organizations that browse the inventory, make reservations, and receive booking confirmations.

Secondary Users:

• Manager – Oversees business performance, approves or edits bookings, and manages product listings.

Context of Use

Rental staff and management will access the system using office desktops or tablets. Customers will interact with the system via a responsive website on their smartphones, tablets, or laptops, allowing them to book items and make inquiries remotely.

2. Requirement & Functionality

Here are the main features we want our project to have:

- FR-1: Users should be able to browse a categorized catalog of rental items (like furniture and decor) with high-quality images, descriptions, and prices.
- FR-2: Users should be able to check if an item is available by selecting event dates and seeing the results right away.
- FR-3: Users should be able to fill out an online form to make a booking request, including all the details about the event (like the date, location, and the item they want).
- FR-4: Administrators should be able to view and manage bookings through a secure dashboard, where they can accept or decline requests.
- FR-5: Users should receive automated email confirmations when they successfully book an item.
- FR-6: Users should be able to filter rental items by category (like chairs or tables), price range, or event type (like weddings or corporate events).
- FR-7: Users should be able to see a gallery of past events to showcase our client's services and inspire them.

Non-Functional Requirements

These requirements make sure the system is reliable, easy to use, and follows industry standards:

- **Performance** – The system should load pages in less than 2 seconds for 95% of users on a 4G network so that it's responsive.
- **Security:** The system should use HTTPS encryption for all data transfers and have two-factor authentication for administrator access.
- **Accessibility** – The system should follow WCAG 2.1 Level AA standards so that it works with screen readers and keyboard navigation.

- **Usability**: Users should be able to make a booking request in fewer than three clicks from the homepage.
- **Scalability:** The system should be able to handle up to 1,000 concurrent users during peak event seasons without slowing down.
- **Reliability:** The system should be up and running 99.9% of the time, except for scheduled maintenance periods.

Prioritization Table (MoSCoW)

The MoSCoW method prioritizes requirements to align with project scope and timeline

Requirement	Priority	Justification
FR-1, FR-2, FR-3	Must Have	Essential for core functionality: browsing. Checking availability and booking rentals.
FR-5, FR-6	Should Have	Enhances user experience through confirmation emails and filtering capabilities.
FR-4, FR-7	Could Have	Admin dashboard and event gallery improve usability but can be phased in later sprints.

Preliminary Use-Case Table

The use-case table outlines key interactions between actors and the system

Use Case	Description	Actors
Browse Catalog	Users view a categorized list of rental items with filters for category, price, or event type.	Client, Event Planner
Check Availability	Users input event dates to check available rental items in real time.	Client, Event Planner
Submit Booking	Users complete a form requesting rentals, specifying event details and items.	Client, Event Planner

Manage Bookings	Admins access a dashboard to view, approve, or reject booking requests.	Admin
Receive Confirmation	Users receive an automated email confirming booking submission.	Client, Event Planner
View Event Gallery	Users browse a gallery of past events to visualize our client's services.	Client, Event Planner
Submit Inquiry	Users send general inquiries via a contact form for nonbooking requests.	Client, Event Planner

3. Architecture and Components

System Decompositions

This includes the **Presentation Layer, Application Layer, Data Access Layer,** and **Database Layer.**

1. *Presentation Layer (UI/UX):* This layer will be responsible for user interaction, displaying information, and capturing user input. It will provide the c graphical user interface (GUI) and handle all user-facing functionalities.

2. *Application Layer or Business Logic:* This layer will contain the core business rules, workflows, and orchestrate interactions between the presentation and data layers. It will be responsible for processing user requests, validating data, and implementing the system's functionalities.

3. *Data Access Layer (DAL):* This layer will abstract the underlying data storage mechanism. It will provide methods for retrieving, storing, updating, and deleting data without exposing the database's complexities to the application layer. This allows for easier database migration or changes without impacting the higher layers.

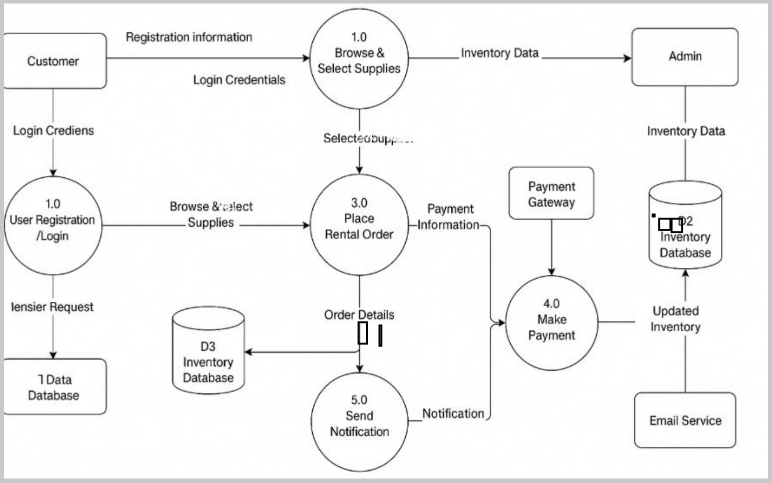
4. *Layer (Data Storage):* This layer will consist of the actual database management system (DBMS) and the stored data. It will be responsible for data persistence, integrity, and security.

Justification for Layered Architecture:

- **Separation of Concerns:** Each layer has a distinct responsibility, making the system easier to understand, develop, and maintain.
- **Modularity and Reusability:** Components within each layer can be developed and tested independently and potentially reused in other parts of the system or even other applications.
- **Scalability:** Individual layers can be scaled independently based on their specific demands (e.g., adding more web servers for the presentation layer, or more database servers for the data layer).
- **Maintainability:** Changes in one layer have minimal impact on other layers, simplifying updates and bug fixes.
- **Testability:** Each layer can be tested in isolation, improving the overall quality of the system.
- **Flexibility:** Allows for easier adoption of new technologies within specific layers without affecting the entire system.

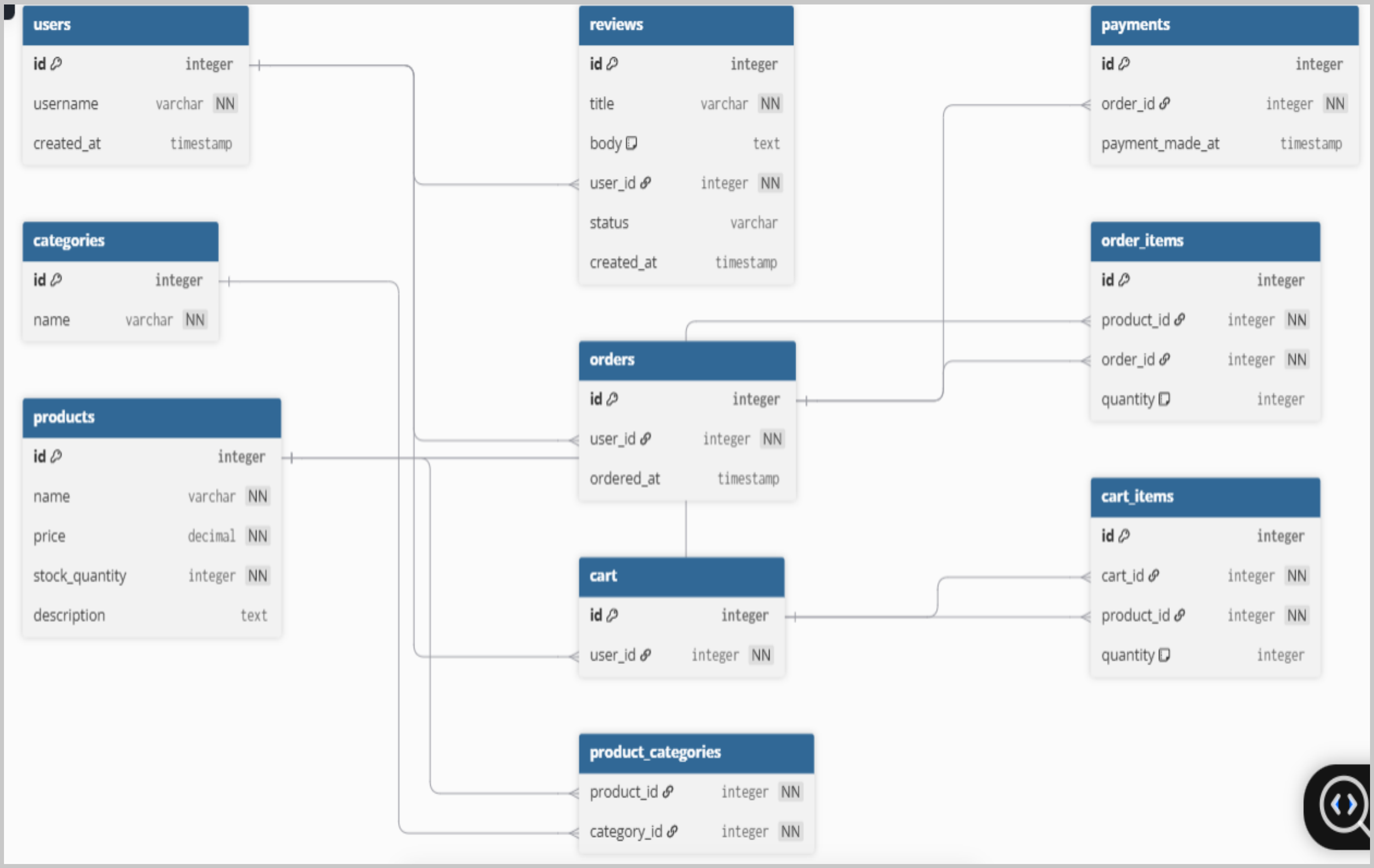
Primary Module

For the primary module, this is our data-flow diagram for our **Event Supply Rental System**:

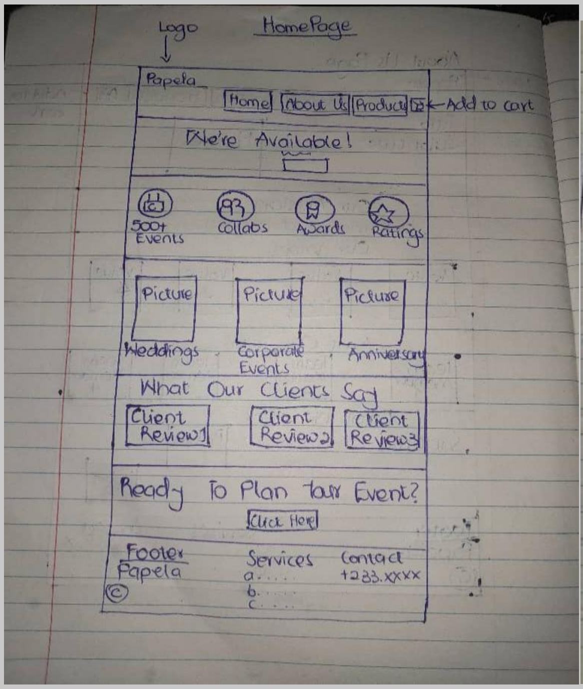


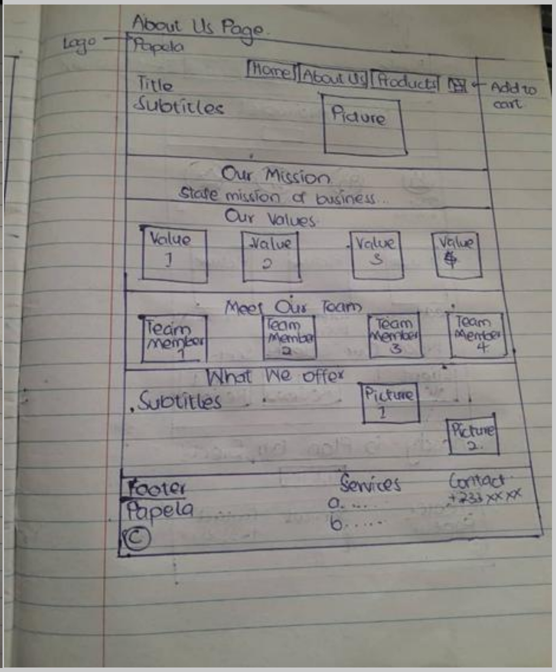
- Data Model

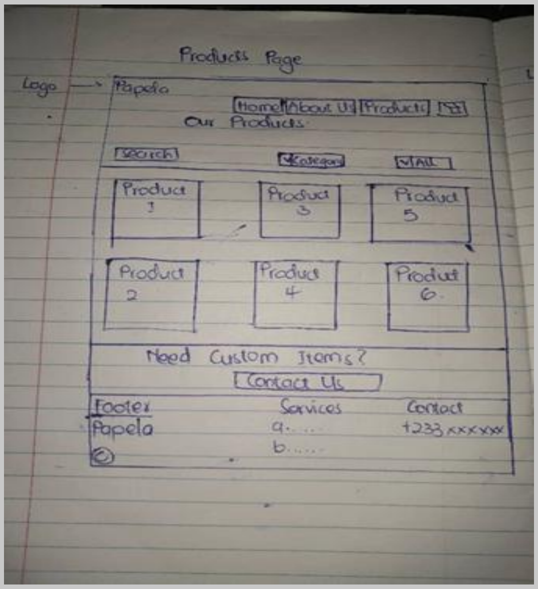
Our project we are currently working on is an **Event Supply Rental System**. This is a diagram of our **Entity Relation Diagram (ERD)**. Below is a link to the diagram with the code attached to it. https://dbdiagram.io/d/208-Database-Design-68794846f413ba35086ef29b



Interface Mock-up







4. Scope, Deliverables & Plan

Deliverable list:

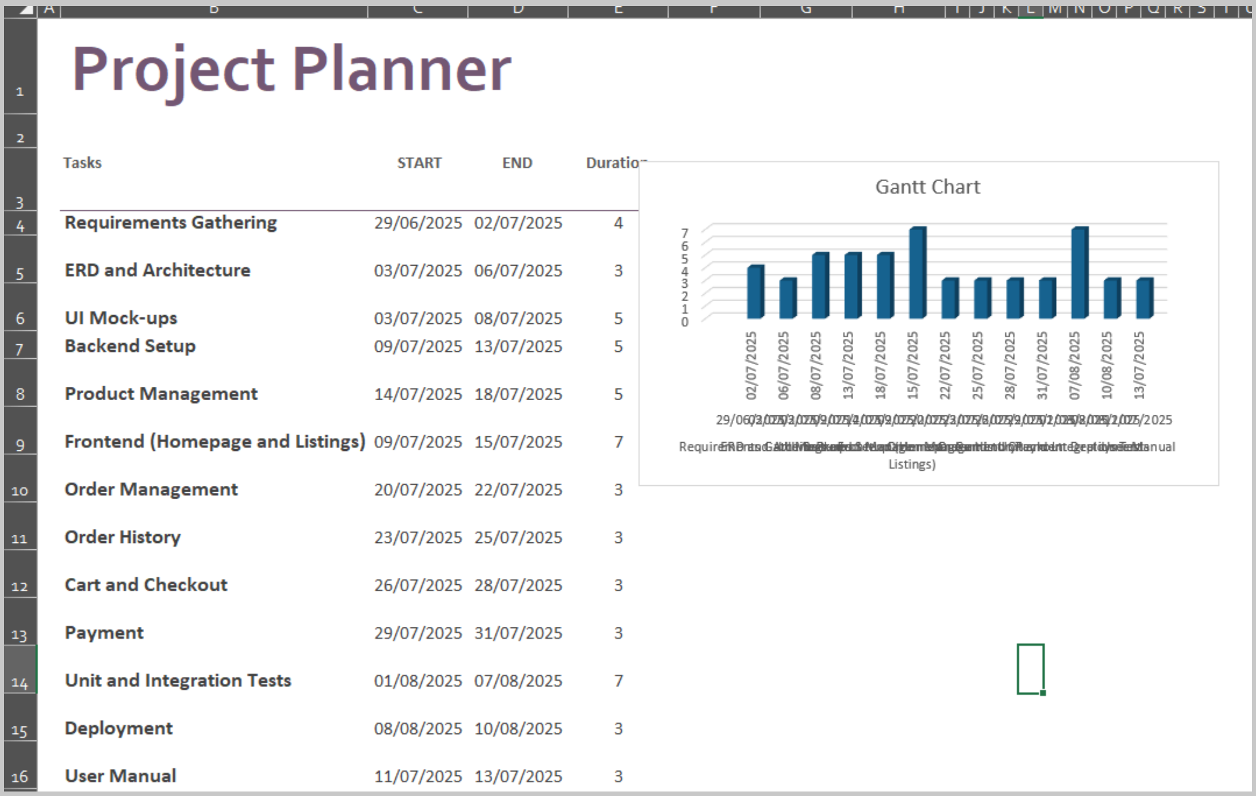
Our project is distributed across three sprints.

- Sprint 1 is the design phase. This involves documenting requirements, creating UI mock-ups, and an ERD model.
- Sprint 2 is the implementation phase. This is where our designs are implemented in code.
- Sprint 3 involves advanced features and integration testing.
- Then there is the final demo where polishes are done, and the project is presented.

Sprint	Deliverable	Notes
1	Project proposal document	Includes requirements, timelines and architecture
1	UI mock-ups	Sketches of the interface
1	ERD model	Shows the database design
2	Backend Setup	Authentication and user roles
2	Product Management	Admin can add, delete and edit products.
2	Homepage and product listing pages (Frontend)	Users can view available products.
3	Order management	Admin can manage orders
3	Order history	Customers can view past orders.
3	Cart and checkout	Customers can add or remove products from cart and initiate checkout.
3	Payment	Simulate payment process
3	Unit and Integration Tests	Conduct tests for backend routes and database integration.

Final demo	Deployment	Hosted version on GitHub pages
Final demo	User Manual	Setup user instructions

Timeline:



Project Planner

Tasks	START	END	Duration
Requirements Gathering	29/06/2025	02/07/2025	4
ERD and Architecture	03/07/2025	06/07/2025	3
UI Mock-ups	03/07/2025	08/07/2025	5
Backend Setup	09/07/2025	13/07/2025	5
Product Management	14/07/2025	18/07/2025	5
Frontend (Homepage and Listings)	09/07/2025	15/07/2025	7
Order Management	20/07/2025	22/07/2025	3
Order History	23/07/2025	25/07/2025	3
Cart and Checkout	26/07/2025	28/07/2025	3
Payment	29/07/2025	31/07/2025	3
Unit and Integration Tests	01/08/2025	07/08/2025	7
Deployment	08/08/2025	10/08/2025	3
User Manual	11/07/2025	13/07/2025	3

Definition of Done:

A deliverable is considered ‘done’ when the following criteria are met:

- **Functionality**: The feature or module meets all specified requirements and acceptance criteria.
- **Code Completion**: All code is written, peer-reviewed, and merged into the main branch with no critical bugs.
- **Testing**: The feature passes unit tests and integration tests and has been manually tested for edge cases.

- **Documentation**: Necessary documentation (e.g., API docs, user instructions, database schema notes) is updated.
- **UI/UX Review**: For front-end features, the UI has been reviewed and aligns with approved mock-ups.
- **Client/PO Review**: The product owner or client has reviewed and accepted the deliverable (for sprint demos).
- **Deployment Ready**: The feature is deployable in the staging environment without errors.

5. Communication, Visibility & Risk

1. Client Touchpoints: It outlines the methods and frequency of communication with clients.

Weekly demo

- Showcase completed features
- Solicit feedback and adjust backlog

Slack/WhatsApp channel

- Real-time updates and quick questions
- Set response windows to prevent late-night pings.

- Monthly summary report

- Progress vs. plan, key metrics, upcoming scope
- Shared as a PDF

These touchpoints help us to keep clients informed and engaged.

2. Team Comms: This refers to the communication strategies within the team. It includes:

Stand-up frequency: Our team holds regular stand-up meetings to find out what each team member accomplished that day with respect to the project. We also talk about obstacles and future tasks.

- Daily 10-minute check-in

- Focus on yesterday's wins, today's plan, blockers

Board link: The team currently uses Notion for our project board.

https://www.notion.so/Project-Management-21802fec7c638053bc34eaba1e04b61c?source=copy_link

3. Risk Log (excerpt):

A risk log is a document or tool used to identify, assess, and manage risks associated with a project. The excerpt include:

- Probability

- Impact

- Mitigation strategies

Risk ID	Description	Probability (1-5)	Impact (1-5)	Score (PxI)	Mitigation
R1	Key resource unavailable	3	3	9	Crosstrain members; maintain backlog buffer
R2	Integration with the legacy system fails	4	5	20	Prototype integration in Sprint 1; allocate spike in Sprint 2
R3	Scope creep from new feature requests	2	4	8	Enforce change control process; revisit priorities weekly

6. Development Process & Compliance

Chosen Process: Scrum with 10-day sprints

Team Nova is using the scrum method to guide development. There is a very limited time in making a functioning software so using a scrum is ideal in speeding up development. Sprints allow us to make any changes our client requires after she has evaluated the features developed at the end of each sprint. Using a scrum will allow our team to develop at a rapid rate and help us to be flexible when dealing with client requirements.

DevSecOps Guard-rails:

This point discusses security measures integrated into the development and operations process.

Branch Protection: A mechanism to prevent unauthorized changes to code branches.

CI Test Coverage Gate: Ensures that code changes are thoroughly tested before being merged.

Secret Scanning: A process to detect and prevent sensitive information (like API keys or passwords) from being exposed in code.

b. Team members are allowed to use generative AI tools such as ChatGPT for tasks such as:

- Debugging for common errors.
- Creating templates or drafting documents.

However, all AI-generated content must be reviewed by Abena Twumwaa Ankobea and Derrick Pemboni. Each member must be able to explain any AI-generated content used in the project. This will ensure accountability, understanding, and integrity.