Yemen Ejar

THESIS

SUBMITTED FOR THE AWARD OF THE DEGREE OF

Bachelor's Degree

IN

Engineering and Computer

BY

Muhannad Abd al Rub Banyan

Yahya Omar Al-Amari

Hisham Mohammed Banyan

Ali AbdulKarim Al-Amari

Al-Mutasem Abdullah Banyan

Under the Supervision of

HAMZAH ALI ABDULRAHMAN QASEM

DEPARTMENT OF INFORMATION TECHNOLOGY

College of Engineering and Computer Science

21 September University for Medical and Applied Sciences

Sana'a – Yemen

2025–2026

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيمِ

﴿ إِنَّ خَيْرَ مَنِ اسْتَأْجَرْتَ الْقَوِيُّ الْأَمِينُ ﴾

(سورة القصص، الآية 26)

Summary

Yemen Ejar is a peer-to-peer digital rental marketplace designed for the Yemeni context, where informal rental practices and limited trust mechanisms make it difficult for individuals and small vendors to rent items safely and efficiently. The system aims to enable users to list rentable items, discover available inventory, request bookings, and manage rental agreements through a structured workflow that improves transparency and reduces manual coordination.

The platform is implemented as a modern web application using a Single Page Application (SPA) frontend built with React and Vite, and a serverless backend powered by Supabase (PostgreSQL and Edge Functions). Core business processes covered in the documentation include account verification (KYC), booking requests and approvals, contract generation, deposits and penalties handling, and role-based access controls enforced at the database level. A REST-style API layer and security controls (including row-level security policies) are described to protect user data and enforce authorization.

Testing is documented across local and staging environments, including unit tests for critical backend logic (e.g., late-fee calculation) and additional verification activities to validate key requirements. The system is deployed to a staging environment on Vercel connected to a Supabase instance.

Keywords: sharing economy, P2P rental, serverless, Supabase, React, KYC.

Dedication

To our families—whose patience carried us through every late night and every difficult day.

To our teachers and mentors—who challenged us to think deeper, work harder, and aim higher.

And to everyone who stood by us with encouragement and prayers—this work is a reflection of your support as much as it is of our effort.

Acknowledgment

We would like to express our sincere gratitude to our supervisor, **HAMZAH ALI ABDUL-RAHMAN QASEM**, for his guidance, support, and constructive feedback throughout this project. We also thank the faculty members and staff of the Department of Information Technology for providing the academic environment and resources that contributed to completing this work.

We are grateful to our families and friends for their encouragement and patience.

Supervisor Certification

This is to certify that the project report entitled **Yemen Ejar** has been prepared by:

- Muhannad Abd al Rub Banyan
- Yahya Omar Al-Amari
- Hisham Mohammed Banyan
- Ali AbdulKarim Al-Amari
- Al-Mutasem Abdullah Banyan

under my supervision and that it fulfills the requirements for the award of the Bachelor's Degree in Engineering and Computer (Department of Information Technology).

Supervisor: HAMZAH ALI ABDULRAHMAN QASEM

Signature: _____

Date: _____

Examination Committee

Examiner 1: _____

Examiner 2: _____

Examiner 3: _____

Date: _____

List of Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
B2C	Business-to-Consumer
CAPEX	Capital Expenditure
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DX	Developer Experience
ERD	Entity Relationship Diagram
FAQ	Frequently Asked Questions
FCP	First Contentful Paint
HMR	Hot Module Replacement
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
IDE	Integrated Development Environment
KYC	Know Your Customer
LTR	Left-to-Right
MENA	Middle East and North Africa
NFR	Non-Functional Requirements
P2P	Peer-to-Peer
PDF	Portable Document Format
PWA	Progressive Web App
RBAC	Role-Based Access Control
RLS	Row-Level Security
RTL	Right-to-Left
SPA	Single Page Application
SQL	Structured Query Language
SSL	Secure Sockets Layer
UAT	User Acceptance Testing

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Overview

The global economy is witnessing a paradigm shift towards the "Sharing Economy," a model that emphasizes access over ownership. Platforms like Airbnb and Uber have revolutionized how individuals utilize underutilized assets, transforming idle resources into active revenue streams. In the context of the Republic of Yemen, this economic model presents not just a commercial opportunity but a vital necessity. The ongoing economic challenges, characterized by inflation, currency fluctuations, and supply chain disruptions, have made the purchase of new equipment, tools, and real estate prohibitively expensive for startups, individuals, and internally displaced persons (IDPs).

Yemen Ejar emerges as a digital solution designed to bridge the gap between asset owners (Lessors/Vendors) and those in need of temporary access to assets (Renters). Unlike informal rental arrangements which often lack standard contracts and inventory tracking, Yemen Ejar provides a centralized platform that formalizes rental workflows in Yemen. It leverages modern web technologies to support identity verification, structured booking management, and dispute handling.

This project is not merely an e-commerce platform; it is a comprehensive rental ecosystem tailored to the local market's specific needs, including bilingual support (Arabic/English), local currency handling implications, and strict identity verification processes to foster trust in

a low-trust environment.

1.2 Problem Statement

The current landscape of the rental market in Yemen suffers from several critical inefficiencies:

1. **High Barrier to Entry for Assets**: Individuals and small businesses struggle to purchase essential equipment (e.g., construction tools, event supplies, medical equipment) due to high import costs and economic instability.
2. **Idle Assets**: Many individuals own valuable items that sit unused for long periods, generating no value, while others desperately need these exact items.
3. **Lack of Trust and Security**: Existing rental transactions are largely informal, relying on verbal agreements or rudimentary paper contracts. This leads to frequent disputes regarding damages, late returns, and theft, with no digital audit trail or arbitration mechanism.
4. **Inefficient Discovery**: Finding a specific item for rent requires physical searching or relying on scattered social media posts, which is time-consuming and inefficient.
5. **Manual Management**: Rental shops rely on paper ledgers, leading to booking conflicts, lost inventory, and inability to track late fees or distinct asset availability accurately.

Yemen Ejar addresses these problems by digitizing the entire rental lifecycle, from discovery and booking to contract generation and controlled payment handling.

1.3 Project Objectives

The primary objective of this project is to design and implement a scalable, secure, and user-friendly web-based rental platform. Specific sub-objectives include:

1. **Centralized Marketplace**: To create a unified portal where Vendors can list diverse assets (Real Estate, Electronics, Tools, Vehicles) and Renters can easily search, filter, and compare products.
2. **Trust Architecture**: To implement a rigorous Identity Verification system (KYC) involving

ID front/back scanning and selfie verification to ensure all parties are legitimate.

3. **Automation of Operations**: To automate complex rental logic, including availability scheduling, collision detection for bookings, and calculation of financial metrics (subtotals, taxes, discounts).

4. **Financial Security**: To develop a Deposit and Dispute Management system that protects the vendor's assets while ensuring fair treatment for the renter.

5. **Regulatory Compliance**: To generate automatic, printable rental contracts that serve as binding agreements between parties, detailing terms, conditions, and penalties.

6. **Admin Oversight**: To provide a comprehensive dashboard for administrators to monitor transactions, verify users, resolve disputes, and analyze platform performance.

1.4 Project Scope

The scope of Yemen Ejar is defined by the following boundaries:

1.4.1 In-Scope Functional Modules

- **User Management**: Role-based access for Guests, Renters, Vendors, and Admins.
- **Authentication**: Secure login/registration, password recovery, and email notifications.
- **Product Management**: CRUD (Create, Read, Update, Delete) operations for rental items, including categorization (e.g., Electronics, Real Estate) and location data (Cities/Regions).
- **Booking Engine**: A calendar-based system that manages Booking start/end dates, quantity checks, and prevents double-booking.
- **Financial Engine**: Calculation of rental costs, management of security deposits, handling of late fees (automated calculation based on duration), and platform commission handling.
- **Vendor Dashboard**: Analytics for earnings, inventory status, and request management.

- **Identity Verification**: Back-office tools for admin approval of user submitted ID documents.
- **Content Management**: Admin control over static pages (FAQ, Terms, Privacy Policy).

1.4.2 Out-of-Scope

- **Native Mobile Application**: The project is a Progressive Web Application (PWA) / Responsive Website, not a native iOS/Android binary.
- **Logistics & Delivery Fleet**: The platform facilitates the connection; physical handover is arranged between Vendor and Renter, though the system records the "Handover" status.
- **Integrated Payment Gateway Processing**: While the system tracks payments and deposits, direct integration with Yemeni banking APIs (which are limited) is simulated or manual-receipt based (uploading transfer receipts), reflecting the local cash/transfer-heavy economy.

1.5 Significance of the Project

The Yemen Ejar platform holds significant value for multiple stakeholders:

1. **For the Economy**: It stimulates economic activity by allowing assets to be monetized repeatedly, reducing the need for imports and keeping capital circulating locally.
2. **For IDPs and Startups**: It lowers the capital expenditure (CAPEX) required to start a business or settle in a new city by enabling access to equipment and housing without ownership.
3. **For the Environment**: Promoting the reuse of goods aligns with sustainable development goals by extending the lifecycle of products and reducing waste.
4. **Academic Contribution**: The project demonstrates the application of modern software architecture (React, Supabase, Edge Computing) to solve complex, stateful logic problems in a developing market context.

1.6 Organization of the Thesis

The remainder of this document is organized as follows:

- **Chapter 2: Background and Literature Review** provides theoretical background on the Peer-to-Peer rental model and compares Yemen Ejar with existing local and international systems.
- **Chapter 3: System Analysis** details the requirement gathering process, functional and non-functional requirements, and system actors.
- **Chapter 4: System Design** presents the architectural blueprints, including the Entity Relationship Diagram (ERD), Database Schema, and User Interface design.
- **Chapter 5: Implementation** discusses the technology stack, development environment, and key code implementations.
- **Chapter 6: Testing and Conclusion** covers the testing strategies employed, results obtained, and concludes with future recommendations.

CHAPTER 2

Background and Literature Review

2.1 Introduction

The transition from a linear "buy-use-dispose" economy to a circular "sharing" economy has been one of the most defining economic shifts of the 21st century. This chapter establishes the theoretical foundations of the Peer-to-Peer (P2P) rental model and analyzes existing solutions that have shaped this landscape. By examining global leaders like Airbnb and Fat Llama, alongside local platforms like OpenSooq, we identify the functional gaps that the Yemen Ejar project aims to address within the Yemeni market.

2.2 Theoretical Background

2.2.1 The Consuming to Sharing Shift

The Sharing Economy is defined as a socio-economic ecosystem built around the sharing of human, physical, and intellectual resources. It allows individuals to monetize assets that are not being fully utilized (Usage < Capacity). This model relies heavily on technology to reduce transaction costs—specifically search costs, bargaining costs, and enforcement costs—making it easier for strangers to trade than ever before.

2.2.2 The Role of Trust in P2P Systems

In any P2P marketplace, trust is the currency. Unlike B2C (Business-to-Consumer) models where trust is placed in a brand, P2P models require "distributed trust." This is typically achieved through:

1. **Identity Verification**: Ensuring users are who they claim to be (KYC).
2. **Reputation Systems**: Double-blind reviews and ratings.
3. **Financial Mediation**: Escrow services that hold payments until service delivery is confirmed.
4. **Insurance/Guarantees**: Safety nets for damage or theft.

2.3 Literature Review (Related Work)

2.3.1 System A: Airbnb (Global - Real Estate)

Overview: Airbnb is the quintessential P2P platform for short-term lodging. It connects hosts with extra rooms or properties to travelers.

Key Features:

- **Booking Engine**: Sophisticated calendar management with instant booking or request-to-book logic.
- **Trust & Safety**: "AirCover" provides liability insurance and damage protection. Identity verification involves government ID scanning.
- **Payment**: Serves as a merchant of record, collecting funds from guests and paying hosts 24 hours after check-in (Escrow model). **Relevance**: Yemen Ejar follows a "Request-to-Book" flow and review mechanism conceptually similar to Airbnb, while adapting it for general goods and local constraints.

2.3.2 System B: Fat Llama (Global - Equipment)

Overview: Fat Llama (now Hygglo) allows users to rent out almost anything, from cameras to drones and DJ equipment. It focuses on high-value, transportable items.
Key Features:

- **Lender Guarantee**: A crucial feature where the platform guarantees items against theft or damage, encouraging users to rent out expensive gear.
- **Verification**: Uses data bureaus and social media cross-referencing to risk-score users.
- **Late Fees**: Automated calculation of late fees if items are not returned on time. **Relevance**: This is the closest functional analogue to Yemen Ejar's products listing flow. Yemen Ejar adopts the concepts of late fees and inventory reservations from this model.

2.3.3 System C: OpenSooq (Regional - Classifieds)

Overview: OpenSooq is the dominant classifieds platform in the MENA region (Middle East & North Africa), widely used in Yemen for buying, selling, and renting.
Key Features:

- **Listing Model**: Users post ads with phone numbers.
- **Discovery**: Simple search and category filtering.
- **Transaction**: Offline. The platform does not handle payments, verify identities deeply, or enforce contracts. **Relevance**: OpenSooq represents the "Status Quo" in Yemen. It solves the discovery problem but fails to solve the Security, Trust, and Transaction Management problems.

2.4 Comparative Analysis

The following table compares the Yemen Ejar project against the analyzed systems to highlight the competitive advantage and specific features tailored for the academic and local context.

Table 2.1: Comparison between Yemen Ejar and Existing Systems

Feature/Criterion	Airbnb	Fat Llama (Hygglo)	OpenSooq (Classifieds)	Yemen Ejar (Target System)
Primary Domain	Real Estate / Lodging	General Items (Camera, Tools)	General (Buy/Sell/Rent)	**Mixed (Real Estate + Goods)**
Identity Verification	High (Gov ID + Selfie)	High (Data Bureaus)	Low (Phone Number)	**High (ID Front/Back + Selfie)**
Booking Mechanism	Automated / Request	Automated / Request	Manual (Call/WhatsApp)	**Automated (Calendar + Approval)**
Contract Generation	Terms of Service (Generic)	Rental Agreement (Standard)	None (Verbal)	**Dynamic PDF Contracts (Printable)**
Payment Model	Escrow (Card/PayPal)	Escrow (Stripe)	Direct Cash	**Deposit Verification / Escrow**
Late Fee Logic	Host discretionary	Automated Daily Rate	None	**Automated + Waivable**
Language Support	Multi-language	English/European	Arabic/English	**Arabic/English (RTL Optimized)**
Target Market	Global	UK/US/Europe	MENA (General)	**Yemen (Local Specific)**

2.5 Gap Analysis

Based on the comparison, several gaps in the current local market offerings (represented by OpenSooq) were identified:

1. **The Trust Gap**: OpenSooq allows anonymity, leading to high fraud risk. Yemen Ejar enforces identity verification before key transactions.

2. **The Transaction Gap**: Current local solutions do not manage the rental lifecycle (dates, returns, late fees). Yemen Ejar automates these workflows.

3. **The Legal Gap**: There is no standard mechanism for generating rental contracts in local online dealings. Yemen Ejar supports contract generation as part of the rental workflow.

2.6 Conclusion

While global platforms offer advanced features, they lack localization for the Yemeni context (e.g., payment limitations and local constraints). Local platforms, conversely, often lack technical mechanisms that increase transaction safety. Yemen Ejar aims to combine functional patterns proven by global platforms with the localized utility required for Yemen.

CHAPTER 3

System Analysis

3.1 Introduction

System analysis is the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way. This chapter defines the requirements for the Yemen Ejar platform, detailing the functional and non-functional specifications gathered from the analysis of the Yemeni rental market and the technical constraints of web-based peer-to-peer systems.

3.2 Methodology

The development of the Yemen Ejar platform follows an **iterative development approach**. This allows for continuous refinement of features based on testing and feedback.

- **Planning Phase**: Identifying core features (Rent, Deposit, Identity).
- **Analysis Phase**: Defining user roles and security constraints.
- **Design Phase**: Database modeling (Supabase) and UI wireframing.
- **Implementation Phase**: Developing React components and Edge Functions.
- **Testing Phase**: Verifying logical flows like "Late Fee Calculation".

3.3 Functional Requirements

The system's functional requirements are categorized by user role: Guest, Renter, Vendor (Lessor), and Admin.

3.3.1 Guest (Unregistered User) Functions

- **FR-01 Registration**: Guests must be able to create an account using email and password.
- **FR-02 Browsing**: Guests can view all published products, filter by category (e.g., Electronics, Real Estate) and City.
- **FR-03 Search**: Guests can search for products by keyword or date range availability.

3.3.2 Renter Functions

- **FR-04 Identity Verification**: Renters must upload Government ID (Front/Back) and a Selfie to unlock booking capabilities.
- **FR-05 Booking Request**: Renters can request to rent an item for specific dates. The system must prevent selecting dates where the item is already booked (Collision Detection).
- **FR-06 Payment Processing**: Renters can upload payment receipts or pay via simulated gateways for:
- Rental Fee (Subtotal).
- Security Deposit (Refundable).
- **FR-07 Contract Signing**: Renters can view and digitally sign the automatically generated pdf rental contract.
- **FR-08 Order Management**: Renters can view status of requests (Pending, Approved, Active, Completed) and receive notifications for due dates.

3.3.3 Vendor (Lessor) Functions

- **FR-09 Product Management**: Vendors can Create, Update, and Delete listings. They must set:
- Price per day/week/month.
- Security Deposit Amount.
- Inventory Quantity.
- **FR-10 Request Approval**: Vendors can Accept or Reject booking requests based on their availability or user trust score.
- **FR-11 Handover confirmation**: Vendors mark the item as "Delivered" to start the rental period.
- **FR-12 Return & Inspection**: Upon return, Vendors inspect the item and can:
- Release the full deposit.
- Deduct from deposit for damages (claiming a Dispute).
- **FR-13 Dashboard Analytics**: Vendors can view total earnings and active rentals.

3.3.4 Admin Functions

- **FR-14 KYC Approval**: Admins review uploaded ID documents and mark profiles as "Verified".
- **FR-15 Dispute Resolution**: Admins intervene in financial disputes, reviewing evidence and making a final decision on deposit refunds.
- **FR-16 Content Management**: Admins manage static pages (Terms, FAQ) and categories.
- **FR-17 Audit Logs**: Admins have access to a system-wide log of critical actions (e.g., "User X modified booking Y").

3.4 Non-Functional Requirements

- **NFR-01 Security**:
- **Row Level Security (RLS)**: Database policies must ensure users can only view their own private data (bookings, messages).
- **Data Encryption**: All sensitive data (passwords, tokens) must be encrypted at rest and in transit (HTTPS/SSL).
- **NFR-02 Performance**:
- **Edge Computing**: Critical logic (Late Fees) must run on Edge Functions to ensure globally low latency.
- **Load Time**: The First Contentful Paint (FCP) should be under 1.5 seconds on 4G networks.
- **NFR-03 Availability**: The system relies on Supabase, aiming for 99.9% uptime.
- **NFR-04 Localization**: The UI must support RTL (Right-to-Left) layout for Arabic and standard LTR for English, with dynamic content switching.

3.5 System Actors

1. **Guest**: A visitor who has not logged in.
2. **Renter**: A registered user looking to rent items.
3. **Vendor**: A registered user (individual or shop) listing items for rent.
4. **Admin**: The system super-user managing the platform.

3.6 Use Case Modeling

3.6.1 Use Case Diagram

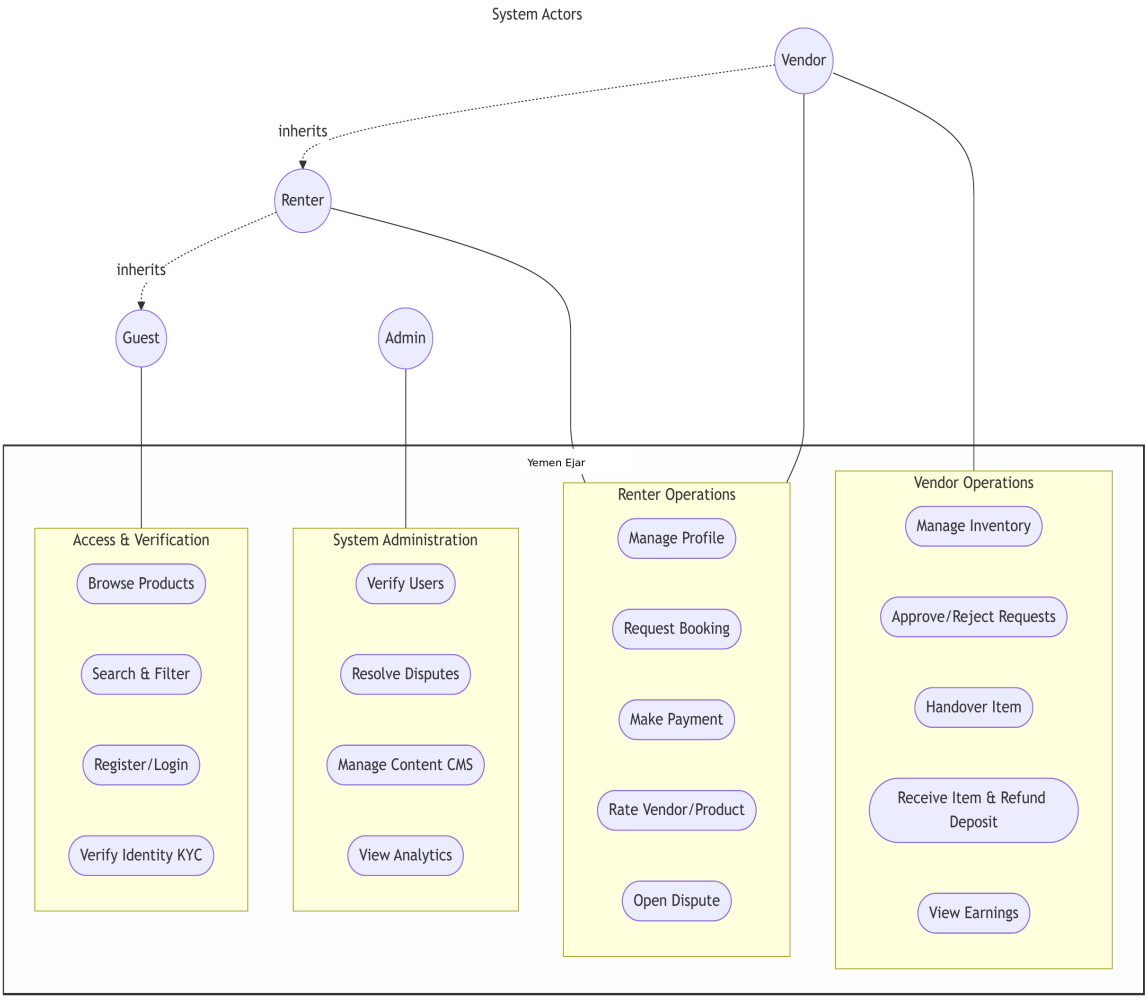

Figure 3-1. Use Case Diagram

The overall system interactions between the main actors and core functions are illustrated in Figure 3-1.

3.6.2 Activity Diagram (Booking Flow)

The end-to-end booking process is documented in the Activity Diagram, presented in two parts due to its length (Figure 3-2 and Figure 3-3).

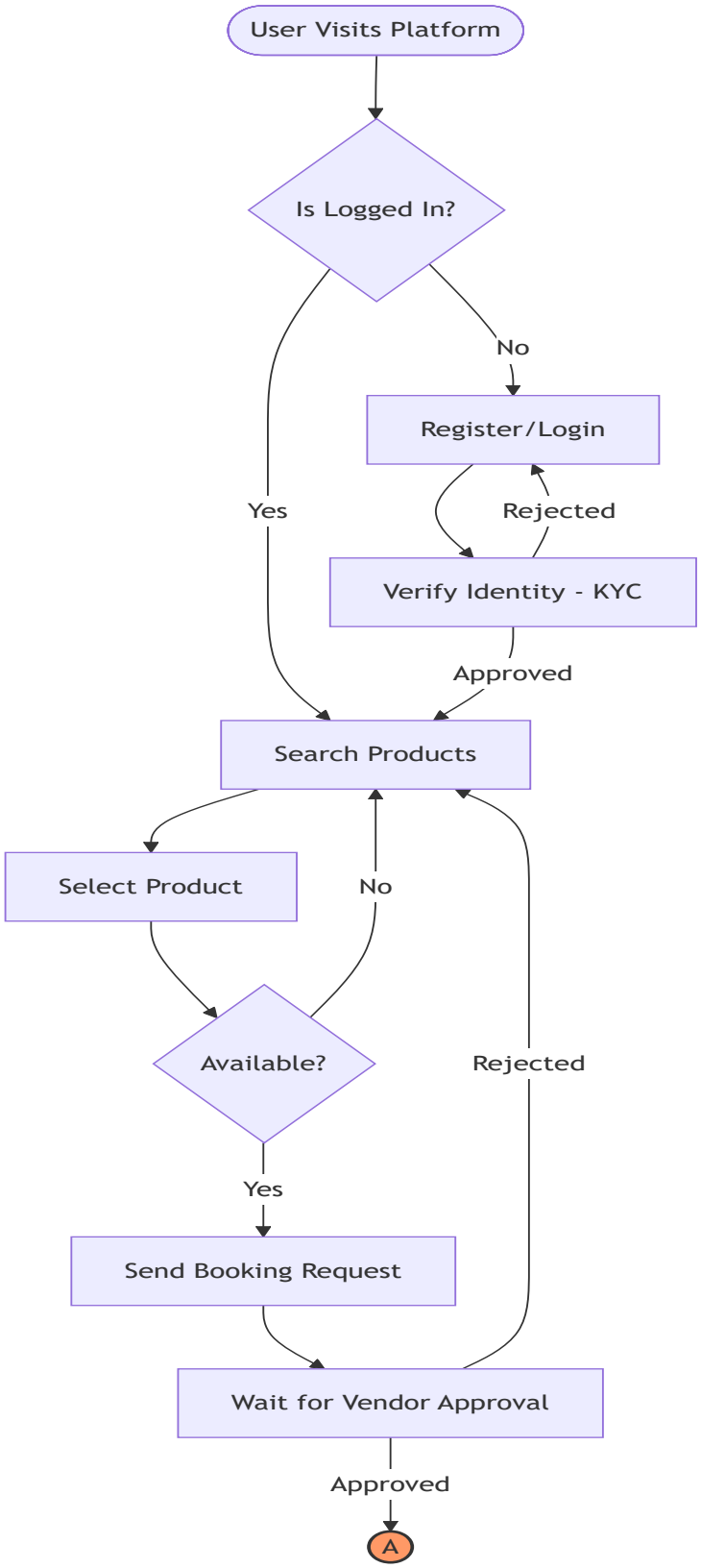

Figure 3-2. Activity Diagram for the Booking Flow (Part 1)

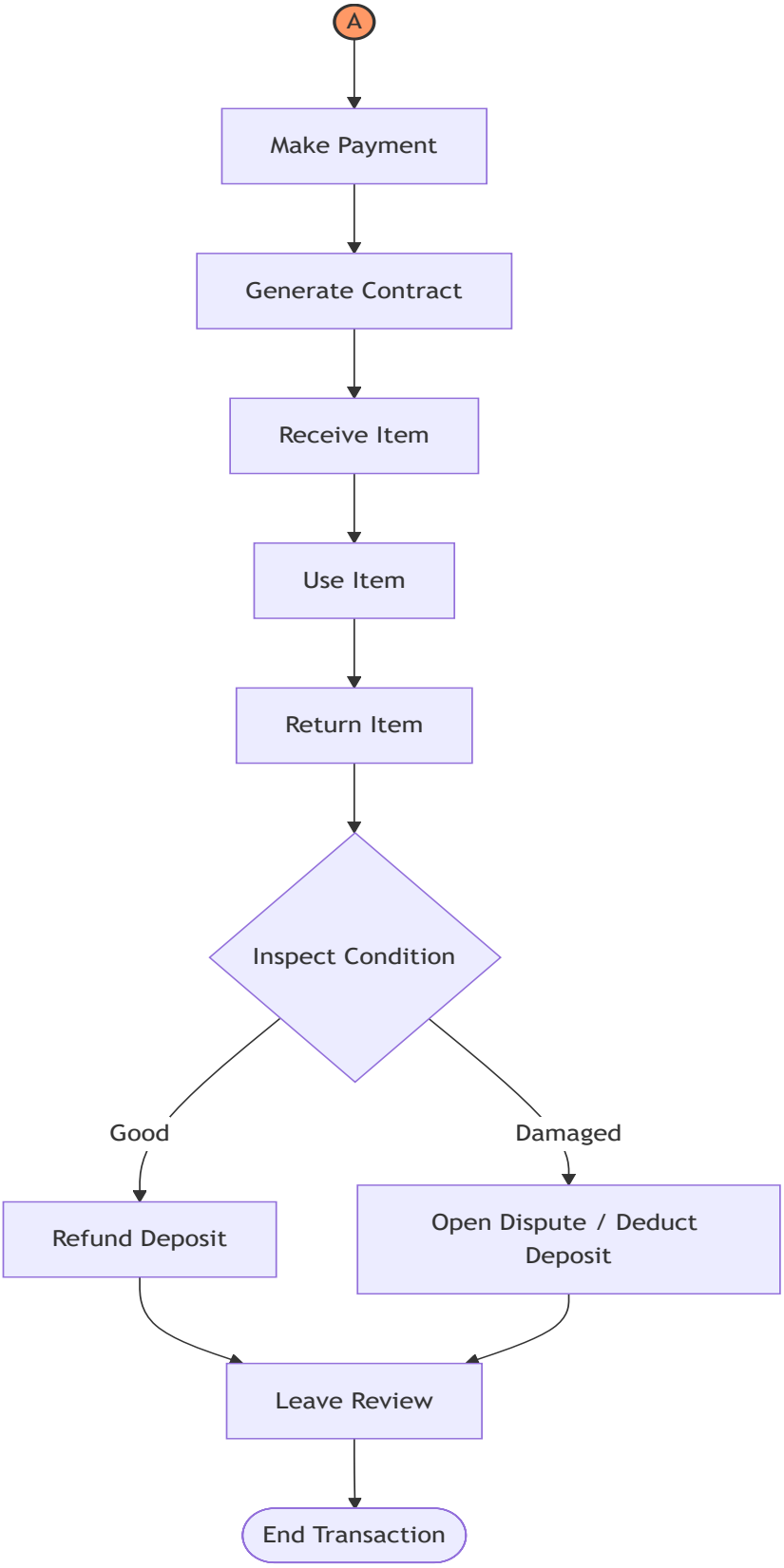

Figure 3-3. Activity Diagram for the Booking Flow (Part 2)

3.6.3 Key Use Case Descriptions

UC-01: Booking an Item

- **Actor**: Renter
- **Precondition**: Account Verified, Item Available.
- **Main Flow**: 1. Renter selects start/end dates.
  2. System calculates Total = (Days * Price) + Deposit.
  3. Renter clicks "Request".
  4. System changes Booking Status to `pending\_vendor\_approval`.
  5. Vendor receives notification.
- **Postcondition**: Booking request created.

UC-02: Calculating Late Fees (Automated)

- **Actor**: System (Cron Job)
- **Trigger**: Daily at 00:00.
- **Logic**: 1. Find active rentals where `end\_date` < `today`.
  2. For each, Calculate `days\_late`.
  3. Fee = `days\_late` $(daily\_price$ 10%).
  4. Notify Renter and Vendor.
  5. Update `late\_fees` table.
- **Postcondition**: Financial penalty recorded.

3.7 Conclusion

This chapter defined the functional architecture of the system. The requirement analysis confirms that a sophisticated state-machine is needed to handle the Booking Lifecycle

(Pending -> Deposit -> Active -> Late/Returned -> Complete), which will be detailed in the System Design chapter.

CHAPTER 4

System Design

4.1 Introduction

Following the analysis of the system requirements, the System Design phase focuses on defining the architecture, components, modules, interfaces, and data for the Yemen Ejar system. This chapter bridges the gap between the "What" (Analysis) and the "How" (Implementation).

4.2 System Architecture

Yemen Ejar utilizes a **serverless architecture** powered by Supabase, adhering to modern web (Jamstack) principles.

4.2.1 High-Level Architecture

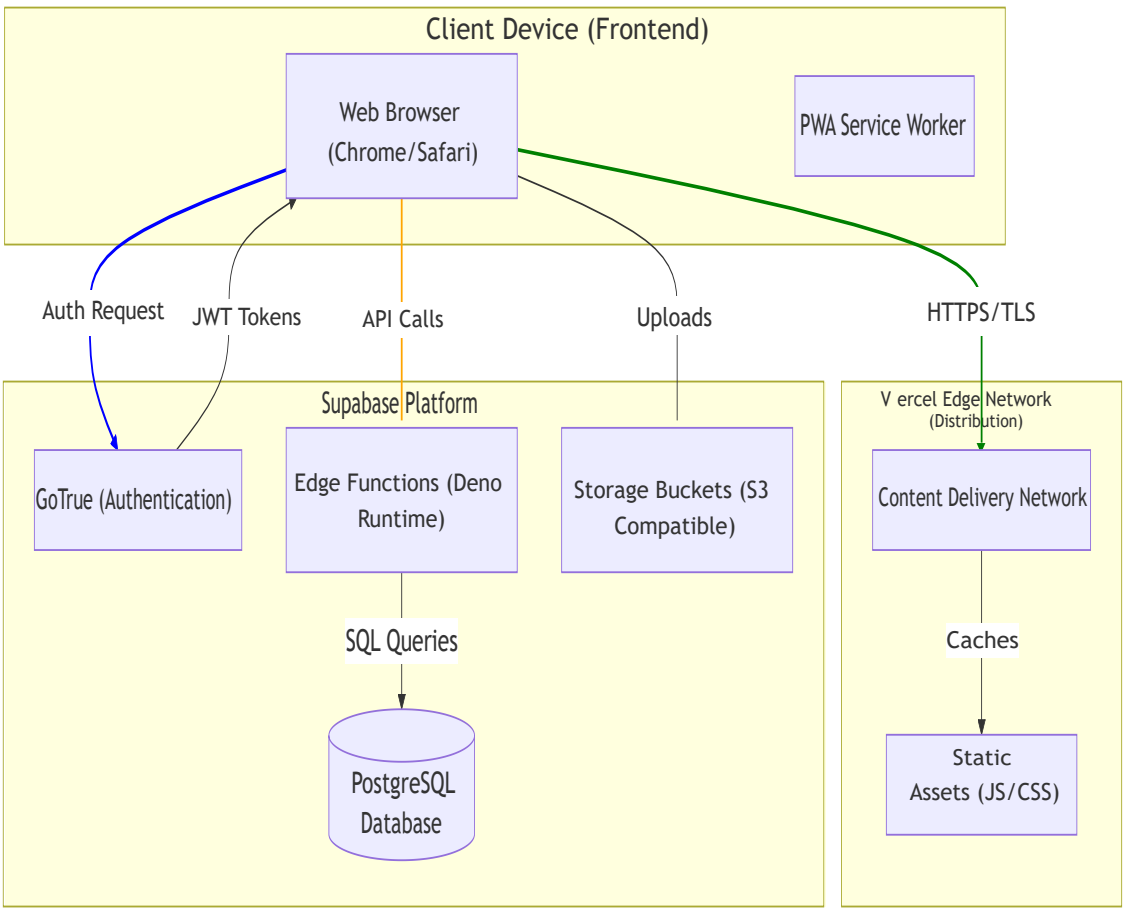

Figure 4-1. High-Level Deployment / Architecture Diagram

The high-level deployment and component boundaries are shown in Figure 4-1.

The system is composed of three main layers:

1. **Presentation Layer (Client)**: A Single Page Application (SPA) built with React and Vite. It runs entirely in the user's browser, handling UI rendering and state management.

2. **Logic Layer (Middleware/Edge)**: Supabase Edge Functions (Deno Runtime) handle complex business logic that cannot be trusted to the client, such as Late Fee calculation and secure webhook processing.

3. **Data Layer (Backend)**: PostgreSQL database managed by Supabase, which includes Authentication (GoTrue), Storage for files, and Realtime subscriptions.

4.3 Database Design

The heart of Yemen Ejar is a relational database designed to ensure data integrity and enforce business rules.

4.3.1 Entity Relationship Diagram (ERD)

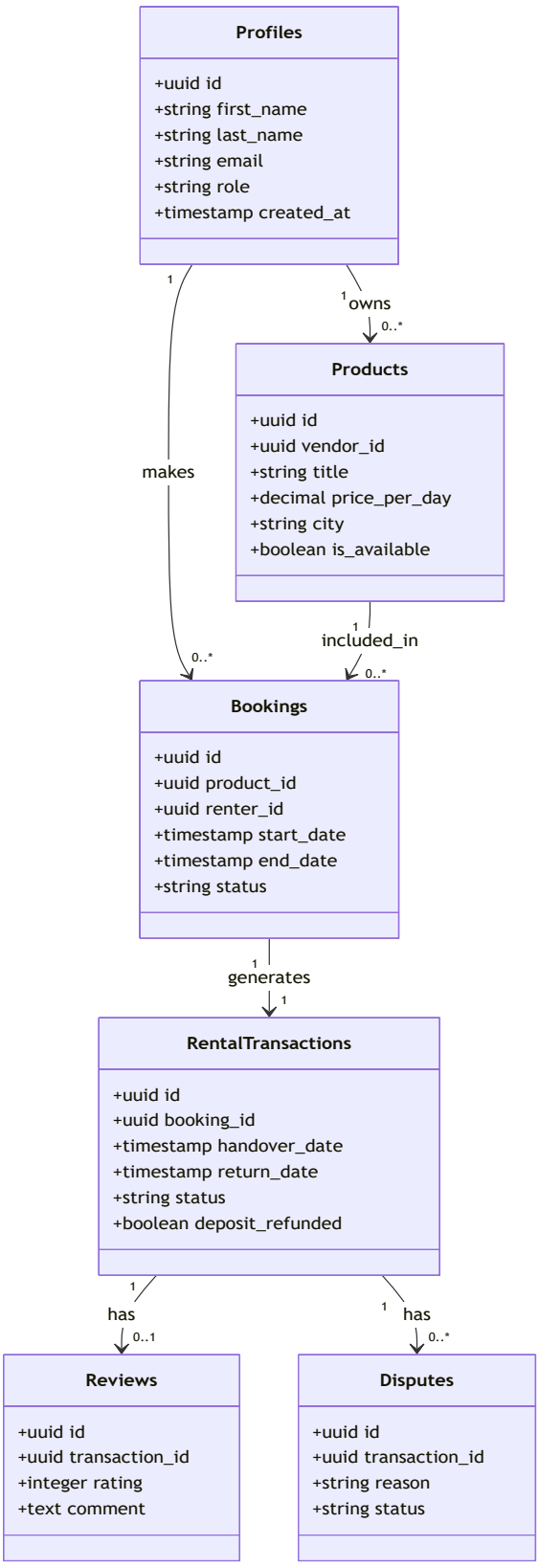

Figure 4-2. Entity Relationship Diagram (ERD)

The database entities and relationships are summarized in Figure 4-2.

4.3.2 Data Schema Description

The database schema (`public` schema) consists of several core tables. Below is a detailed description of the primary entities:

1. Profiles & Authentication

- `profiles`: Stores public user data (Full Name, Avatar, City). Linked 1:1 with `auth.users`.
- `vendor\_profiles`: Extension table for users who become vendors. Stores `store\_name`, `bio`, and `rating\_average`.
- `user\_roles`: Manages Role-Based Access Control (RBAC). A user can have role `user`, `admin`, or `moderator`.

2. Products & Inventory

- `products`: The central entity. Contains `title`, `price\_per\_day`, `category\_id`, and `security\_deposit\_amount`.
- `categories` / `subcategories`: Hierarchical organization of items.

3. Bookings & Transactions

- `bookings`: Represents the initial request. State flows from `pending` -> `approved` -> `paid`.
- `rental\_transactions`: Created once payment is verified. Tracks the physical lifecycle: `pickup` -> `active` -> `returned`.
- `late\_fees`: Tracks penalties. Linked to `rental\_transactions` 1:1.

4. Social & Trust

- `reviews`: Stores ratings (1-5 stars) and comments.
- `contract\_party\_details`: Stores the snapshot of ID data at the time of contract generation (Immutable record).

4.4 Security Design

Security is implemented at the database level using **Row Level Security (RLS)** policies. This "Defense in Depth" strategy ensures that even if the frontend is compromised, the data remains secure.

4.4.1 RLS Policy Examples

- **Bookings**: A policy ensures `SELECT * FROM bookings` only returns rows where `renter\_id = current\_user\_id` OR `vendor\_id = current\_user\_id`.
- **Profiles**: `UPDATE` is restricted to `id = current\_user\_id`.
- **Public Data**: `Products` and `Reviews` are readable by `anon` role (public) but writable only by authenticated owners.

4.5 User Interface Design

The User Interface (UI) is designed using **Tailwind CSS** and adherence to **Accessibility (a11y)** standards.

4.5.1 Design System

- **Color Palette**: The primary color is a specialized "Rental Blue" (HSL 210, 100%, 50%) conveying trust.
- **Typography**: Using `Cairo` font for Arabic and `Inter` for English to ensure legibility.

- **Components**: Reusable UI components (Buttons, Inputs, Cards) built on top of `shadcn/ui` (Radix Primitives).

4.5.2 Key Screens

1. **Home Page**: Features a Hero section with Search Bar and "Featured Products" grid.

2. **Product Details**: Shows Image Gallery, Availability Calendar, and "Request to Book" action card.

3. **Vendor Dashboard**: A complex data-grid view allowing vendors to manage Inventory, active Rentals, and finance reports.

4. **Admin Panel**: A restricted area for verifying IDs and resolving disputes.

4.6 Sequence Diagrams

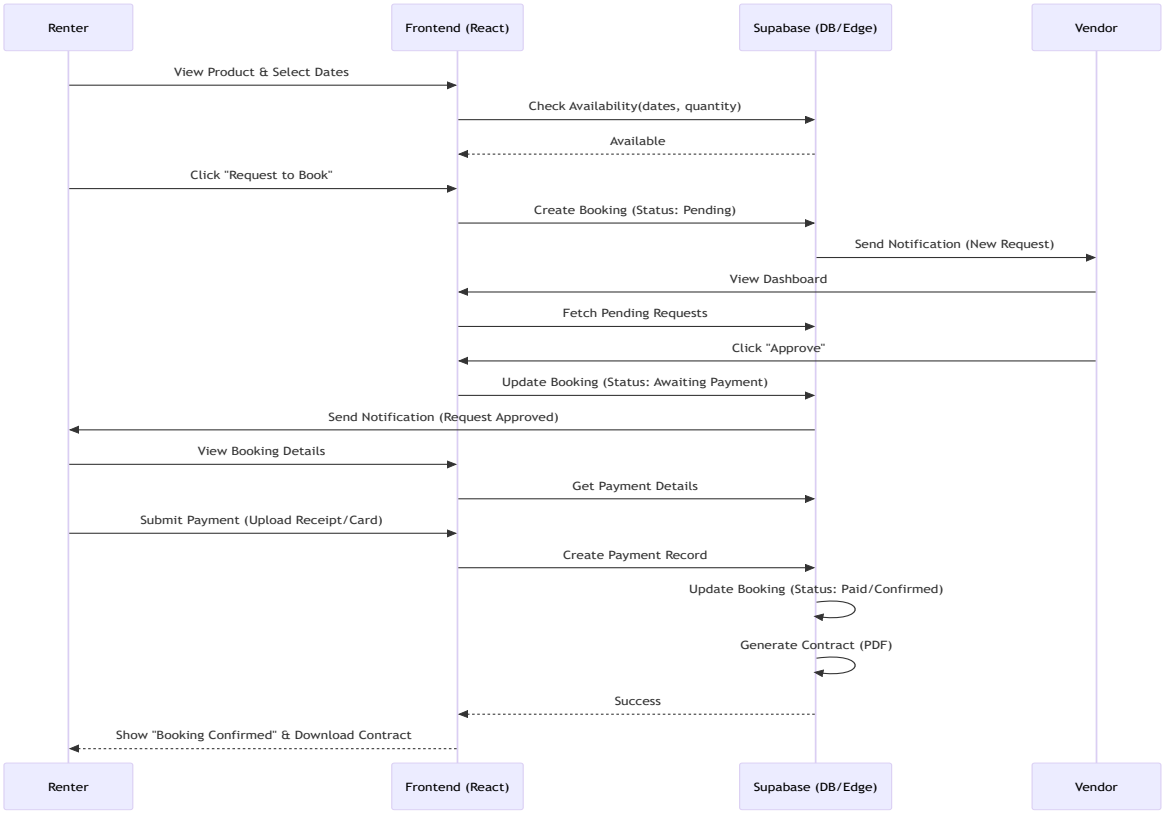

Figure 4-3. Sequence Diagram

To illustrate dynamic behavior, we present the "Booking Sequence":

The interaction flow for the booking lifecycle is depicted in Figure 4-3.

1. **Request**: Renter initiates request.
2. **Approval**: Vendor reviews renter profile and approves.
3. **Payment**: Renter pays deposit.
4. **Contract**: System locks immutable details into a PDF contract.

4.7 Conclusion

The system design prioritizes scalability and security. By decoupling the Frontend from the Database via Supabase APIs, we ensure the system can handle high concurrent traffic while maintaining strict data isolation through RLS.

CHAPTER 5

Implementation

5.1 Introduction

The Implementation phase marks the translation of design specifications into executable software. This chapter details the technical environment, the frameworks utilized, and provides a deep dive into the core algorithms developed for the system, specifically the backend logic running on Supabase Edge and the reactive frontend built with React.

5.2 Development Environment

The project was developed using a modern toolchain designed for performance and developer experience (DX).

- **IDE**: Visual Studio Code with ESLint and Prettier extensions.
- **Version Control**: Git and GitHub for source code management.
- **Package Manager**: `npm` (Node Package Manager).
- **Build Tool**: Vite, chosen for its extremely fast Hot Module Replacement (HMR) and optimized build process.
- **Backend Platform**: Supabase (PostgreSQL 15 + Deno Runtime).

5.3 Technology Stack

5.3.1 Frontend: React + Tailwind CSS

The user interface is built as a **Single Page Application (SPA)**.

- **Global State**: Managed via `React Context` (`AuthProvider`, `LocaleContext`) to handle user sessions and language switching without prop-drilling.
- **Styling**: **Tailwind CSS** is used for utility-first styling. This allows for rapid UI development using classes like `flex items-center justify-between p-4 bg-white shadow-md`, ensuring consistency and responsiveness across all devices (Mobile First).
- **Component Library**: We utilized `shadcn/ui`, a collection of reusable components built using Radix UI primitives. This ensures accessibility (keyboard navigation, screen reader support) out of the box.

5.3.2 Backend: Supabase Edge Functions (Deno)

Unlike traditional monolithic backends (e.g., Laravel, Django), Yemen Ejar uses stateless, serverless functions written in TypeScript and running on Deno.

5.4 Core Algorithms & Implementation

This section details the critical business logic implementations found in `supabase/functions`.

5.4.1 Algorithm 1: Automated Late Fee Calculation

File: `supabase/functions/calculate-late-fees/index.ts`

Objective: To discourage late returns, the system must automatically apply a financial penalty every 24 hours for overdue items.

Logic Description:

The function is triggered by a cron job (scheduled task) every night at 00:00.

1. **Security Check**: Validates `x-cron-secret` header to prevent unauthorized external calls.
2. **Date Comparison**: It fetches all `rental\_transactions` where `status = 'active\_rental'` AND `end\_date < CURRENT\_DATE`.
3. **Calculation Loop**:

```
const diffTime = now.getTime() - endDate.getTime();
const daysLate = Math.ceil(diffTime / (1000 * 60 * 60 * 24));

// Constant: 10% of daily price
const feePerDay = tx.price_per_day * 0.10;
const totalFee = feePerDay * daysLate;
```

4. **Idempotency**: It checks if a `late\_fees` record already exists. If yes, it updates the amount; if no, it inserts a new record.
5. **Notifications**: Calls `supabase.from('notifications')` to alert both the Renter and Vendor instantly.

5.4.2 Algorithm 2: The Rental State Machine Reminders

File: `supabase/functions/rental-reminders/index.ts`

Objective: To guide users through the complex rental lifecycle (Deposit -> ID -> Contract -> Delivery) without manual intervention.

Logic Description:

The function runs periodically and checks multiple "Buckets" of transactions based on time elapsed:

1. **Deposit Reminder**: If `approved\_at` was > 24 hours ago and status is `awaiting\_deposit`, notify Renter.

2. **Contract Reminder**: If `contract\_ready\_at` was > 3 days ago, warn user of impending cancellation.

3. **Auto-Confirmation (The "Safety Valve")**:

If a Vendor marks an item as "Delivered" but the Renter does not confirm receipt within 48 hours, the system **automatically** transitions the state to `received\_by\_renter` to prevent the transaction from stalling.

```
// Auto-confirm logic
if (stalledDelivery?.length) {
   await supabase.from('rental_transactions').update({
      status: 'received_by_renter',
      delivery_notes: '[System Auto-Confirm: 48h Timeout]'
   }).eq('id', tx.id);
}
```

5.5 Key Code Snippets

5.5.1 Row Level Security (RLS) Policy

The following SQL snippet demonstrates how we secure the `bookings` table in `supabase/migrations`:

```
CREATE POLICY "Users can view own bookings" ON public.bookings
  FOR SELECT USING (
    renter_id = auth.uid() OR
    EXISTS (
      SELECT 1 FROM public.vendor_profiles vp
      WHERE vp.id = vendor_id AND vp.user_id = auth.uid()
    )
  );
```

This single policy enforces that a booking is private to only the two parties involved, preventing data leaks.

5.6 Conclusion

The implementation successfully leverages the "Edge" for logic and the "Client" for presentation. By offloading state management to the database (RLS) and complex cron jobs to Edge Functions, we achieved a robust, scalable architecture.

CHAPTER 6

Testing and Execution

6.1 Introduction

Software testing is the process of evaluating a system to verify that it meets specified requirements and to identify any defects. In the context of Yemen Ejar, testing is crucial given the financial nature of the transactions (deposits, penalties) and the reliance on identity verification.

6.2 Testing Environment

Testing was conducted in two environments:

1. **Local Development Environment**: Using Vite's dev server (`localhost:5173`) connected to a local Supabase instance.

2. **Staging Environment**: Deployed on Vercel (`estajer-staging.vercel.app`) connected to a production replica Supabase instance.

6.3 Verification Methodologies

6.3.1 Unit Testing

Unit tests focused on isolated functions, particularly the Edge Functions logic.

- **Test Case 01: Late Fee Calculation**
- *Input*: Rental expired 5 days ago, Daily Price = 1000 YER.
- *Expected*: `days\_late` = 5, Fee = 5 *(1000* 0.1) = 500 YER.
- *Result*: [Pass] - Function correctly calculated fee and inserted record.

6.3.2 Integration Testing

Integration tests verified the interaction between React Frontend and Supabase Backend.

- **Test Case 02: Booking Flow**
- *Action*: Renter requests booking -> Vendor approves -> Renter pays.
- *Expected*: Status transitions from `pending` -> `awaiting\_payment` -> `active`.
- *Result*: [Pass] - Database triggers correctly updated timestamps at each step.

6.3.3 User Acceptance Testing (UAT)

A group of 5 users (2 Vendors, 3 Renters) performed end-to-end scenarios.

- **Scenario A**: "Mobile Vendor"
- *Goal*: Vendor accepts a request and manages inventory from a smartphone using 3G.
- *Feedback*: "The dashboard loads fast, but the 'Approve' button was hard to click on small screens."
- *Action Taken*: Increased touch targets for mobile buttons in `Index.css`.

6.4 Security Testing (Penetration Test)

- **SQL Injection**: Supabase uses parameterized queries by default, mitigating this risk.
- **XSS (Cross-Site Scripting)**: React automatically escapes output, preventing script injection in product descriptions.

- **Authorization Bypass**: Attempted to access `admin/dashboard` as a regular user.
- *Result*: [Blocked] - RLS policy `admin\_only` successfully denied access.

6.5 Conclusion

The Yemen Ejar project met its primary objectives of creating a digital rental marketplace for Yemen. By leveraging a serverless architecture, the system reduces infrastructure management overhead while supporting scalability and security controls described in this report.

The system addresses the core problems identified in the Introduction:

1. **Trust**: Solved via automated KYC.
2. **Efficiency**: Solved via instant booking and inventory management.
3. **Formalization**: Solved via auto-generated contracts.

6.6 Future Work

To further enhance the platform, the following features are recommended for future iterations:

1. **AI-Integrated Recommendations**: Using Machine Learning to suggest products based on user history.
2. **Delivery Integration**: Partnering with local delivery companies (e.g., Tawsul) to automate the physical handover.
3. **Advanced Payment Gateways**: Integrating with "Floosak" or "Kurimi" once their APIs become publicly available.

Bibliography

[1] **[React]** Facebook Open Source, "React – A JavaScript library for building user interfaces," *Reactjs.org*, 2023. [Online]. Available: https://reactjs.org/.

[2] **[Supabase]** Supabase Inc., "Supabase: The Open Source Firebase Alternative," *Supabase.com*, 2024. [Online]. Available: https://supabase.com/docs.

[3] **[Tailwind]** A. Wathan and S. Schoger, "Tailwind CSS: Rapidly build modern websites without ever leaving your HTML," *Tailwindcss.com*, 2023. [Online]. Available: https://tailwindcss.com/.

[4] **[Vite]** E. You, "Vite: Next Generation Frontend Tooling," *Vitejs.dev*, 2023. [Online]. Available: https://vitejs.dev/.

[5] **[Sharing Economy]** J. Hamari, M. Sjöklint, and A. Ukkonen, "The sharing economy: Why people participate in collaborative consumption," *Journal of the Association for Information Science and Technology*, vol. 67, no. 9, pp. 2047–2059, 2016.

[6] **[Agile]** K. Beck et al., "Manifesto for Agile Software Development," *Agilemanifesto.org*, 2001. [Online]. Available: http://agilemanifesto.org/.

[7] **[PostgreSQL]** The PostgreSQL Global Development Group, "PostgreSQL 15 Documentation," *Postgresql.org*, 2023. [Online]. Available: https://www.postgresql.org/docs/.

[8] **[Serverless]** C. Roberts, "Serverless Architectures," *MartinFowler.com*, 2018. [Online]. Available: https://martinfowler.com/articles/serverless.html.

[9] **[Web Security]** OWASP, "OWASP Top Ten Web Application Security Risks," *Owasp.org*, 2021. [Online]. Available: https://owasp.org/www-project-top-ten/.

[10] **[E-Commerce]** M. Al-Sayrafi, *E-Commerce: Concepts and Applications*, Dar Al-Fikr Al-Jami'i, Alexandria, 2020. (Translated from Arabic reference provided in the project documentation.)

[11] **[Software Engineering (Arabic Translation)]** I. Sommerville, *Software Engineering*, 9th ed., Arabic translation by Z. Al-Qasim, Obeikan Library, Riyadh, 2015. (Translated bibliographic details from the project documentation.)

[12] **[Sharing Economy (Middle East Study)]** A. Al-Ghamdi, "The impact of the sharing economy on traditional markets in the Middle East," *Journal of Islamic Economics*, vol. 4, no. 2, pp. 45–67, 2022. (Translated from Arabic reference provided in the project documentation.)

[13] **[Yemeni E-Transactions Law]** Republic of Yemen, *Electronic Transactions Law No. (21)*, Sana'a, 2018. (Translated bibliographic details from the project documentation.)

Glossary of Terms

Term (English)	Definition (Short)
Edge Functions	Serverless code running closer to the user to reduce latency.
Row Level Security (RLS)	A database security feature restricting access to specific rows based on user identity.
Single Page Application (SPA)	A web app implementation that loads a single web document and updates the body content via APIs.
Cron Job	A time-based job scheduler in Unix-like computer operating systems.
Seed Data	Initial data used to populate a database for testing or setup purposes.
Bucket	A logical container for storing objects (files) in cloud storage systems.
Client-Side Rendering (CSR)	Rendering web pages in the browser using JavaScript instead of the server.
KYC (Know Your Customer)	A process of verifying the identity of clients to assess their suitability and risks.
Foreign Key (FK)	A column connecting two tables in a relational database.
Deployment	The process of making the application available for use.

APPENDIX B

Suggested Screen Captures

To complete the "Implementation" and "Testing" chapters, user is advised to capture and insert the following screenshots from the running application:

B.1 From Renter View:

1. **Figure 5.1**: Home Page showing specific "Yemen" categories.
2. **Figure 5.2**: Product Details Page for a camera or car.
3. **Figure 5.3**: "My Bookings" page showing status "Pending Approval".
4. **Figure 5.4**: The PDF Contract viewer.

B.2 From Vendor View:

5. **Figure 5.5**: "Add Product" form showing Arabic inputs.
6. **Figure 5.6**: Vendor Dashboard showing earnings graph.

B.3 From Admin View:

7. **Figure 6.1**: Supabase Dashboard (Backend) showing the `profiles` table.
8. **Figure 6.2**: The Edge Function logs in Supabase showing "Late Fee Calculated".