MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

Computer Engineering Program

CNG 495

CLOUD COMPUTING

FALL 2025
Capstone Project Proposal
SmartRent

Team Members:

Mahlet Bekele- 2643146

Zeeshan Imran - 2640779

Miguel Tunga Mbabazi- 2600195

TABLE OF CONTENT

Cloud-based Smart Property Management & Rental Platform

SmartRent is a web-based property and rental management system that unifies landlords and tenants under one integrated system. The main purpose of the system is to allow tenants to securely log in, pay rent/bills online, view property details, and submit maintenance requests. Landlords can manage multiple properties, track rent payments, view maintenance history, and update request status in real-time.

We plan to achieve the following mentioned functionalities by the end of this project. These important features include user authentication, role-based access (landlord and tenant), rental and utility payments, maintenance requests submission, and a bill payment reminder.
For scalability and privacy, the system will use a multi-tenant SaaS approach, with each landlord's data segregated. The application will include rent reminders, payment reminders, and maintenance records, and provide a transparent, streamlined, and efficient process for all users.

IMPLEMENTATION:

We plan on implementing a full stack project with proper integration of the frontend with the backend and the database. In our system, the user interacts with the buttons for registering, paying rent or other functionalities, these requests are sent in the form of APIs with the backend. It should have a seamless integration and accurately update the house details(occupied/not_occupied) in real-time and the tenant details for that occupied house.

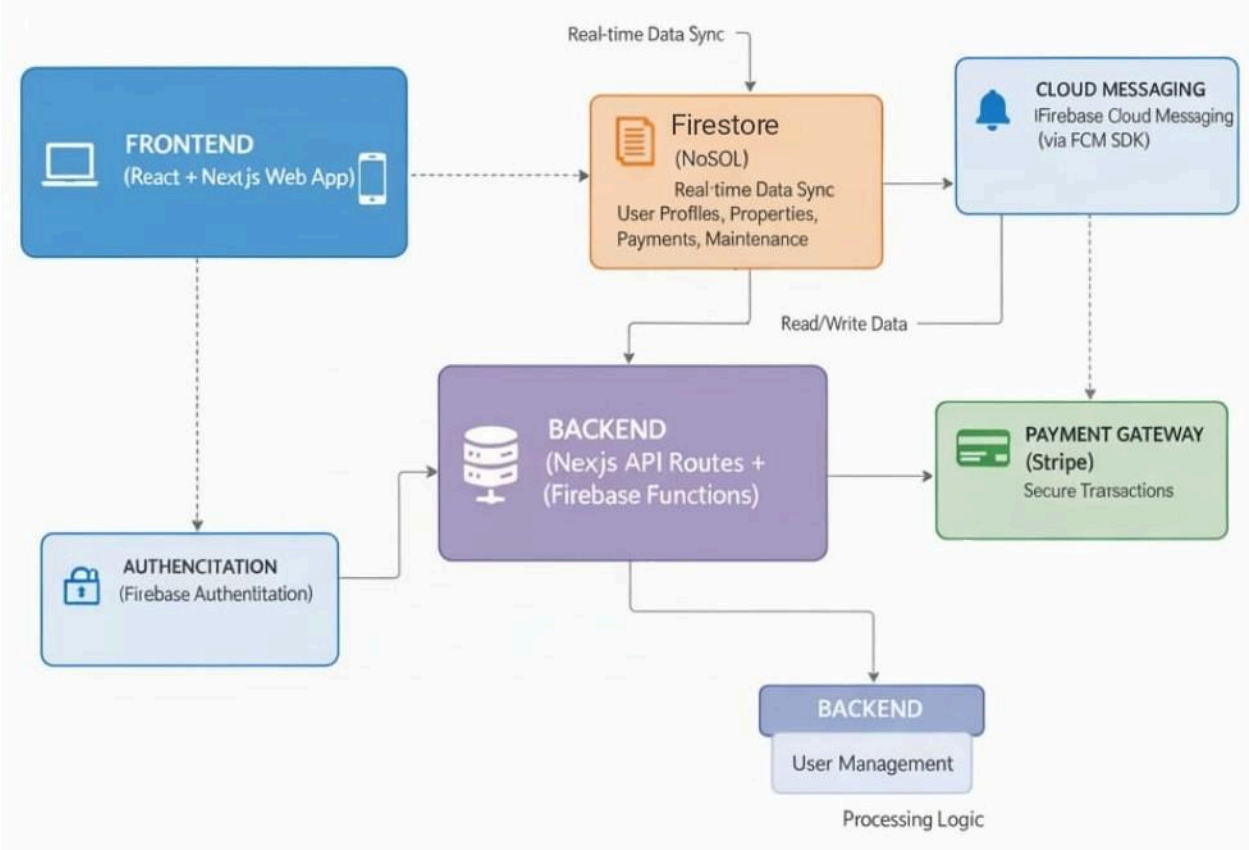

Figure 1: Application Environment Integration flow

Frontend (React + Next.js)

We will build a responsive web frontend for landlords as well as tenants. There would be a separate registration page and login page. The pages we create will be a rent payment dashboard, maintenance request form, reminders and notification page. We'll build the project in React in a Next js framework, which will give us faster loading time, better performance and future scalability for growth.

Backend (Next.js API Routes + Firebase Functions)

Backend would be built through Next js API routes and Firebase Functions to host the server-side code. Next js API gives a straightforward interaction of frontend with the backend server for achieving a Secure authentication and role-based authorization, Payment gateway integration (Paddle/Stripe), request routing for maintenance, and rent reminder system. Firebase is a cloud-based app development platform that provides developers with pre-built backend services. It contains numerous services from authentication, databases, cloud functions, and cloud storage. It is ideal as it enables rapid development and scalability.

Database (Firestore NoSQL)

For data storage and retrieval, we will use Firestore, Firebase's cloud NoSQL database. It's data as collections and documents and is ideal as it offers real-time data synchronization across the clients and the backend and fast development. It will hold user profiles, property details, payments, maintenance requests, and payment status. It handles large volumes of data well.

CLOUD DELIVERY MODELS:

1. **SaaS**

Stripe / Paddle

Provides integrated secured payment processing functionality via cloud APIs.

2. **PaaS**

Firebase (Firestore, Cloud Messaging, Hosting, Authentication):

Provides backend and platform features like real-time database, push, secure authentication, and hosting for web. We can focus on app development while Firebase takes care of infrastructure, scalability, and servers.

Vercel / Render

Hosts React/ Next.js frontend and API routes with automatic building, deployment, and future scaling. Offers a managed platform environment in which we are not required to configure it or manage servers.

DIAGRAMS:

Use Case diagram

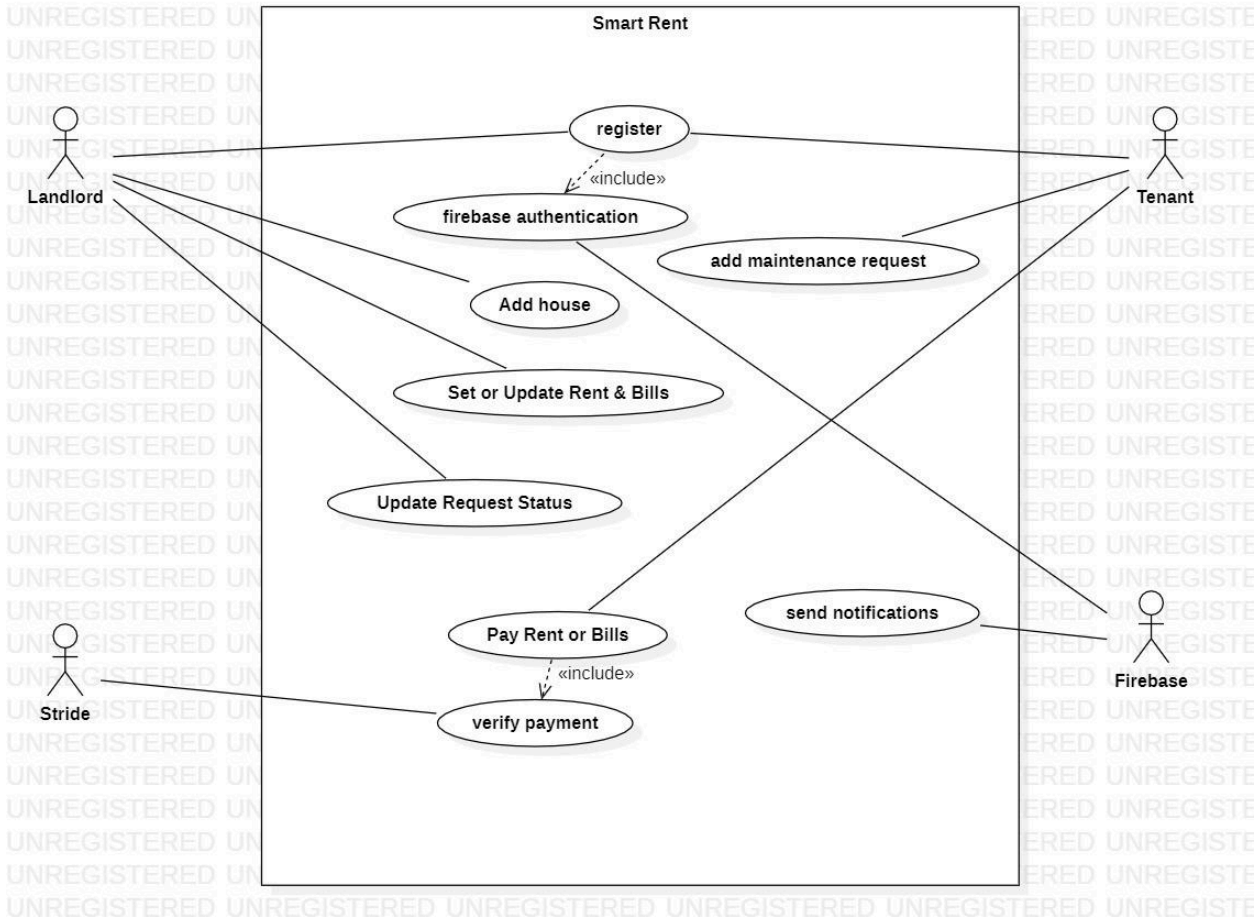

Figure 2: UseCase Diagram

Data flow diagrams

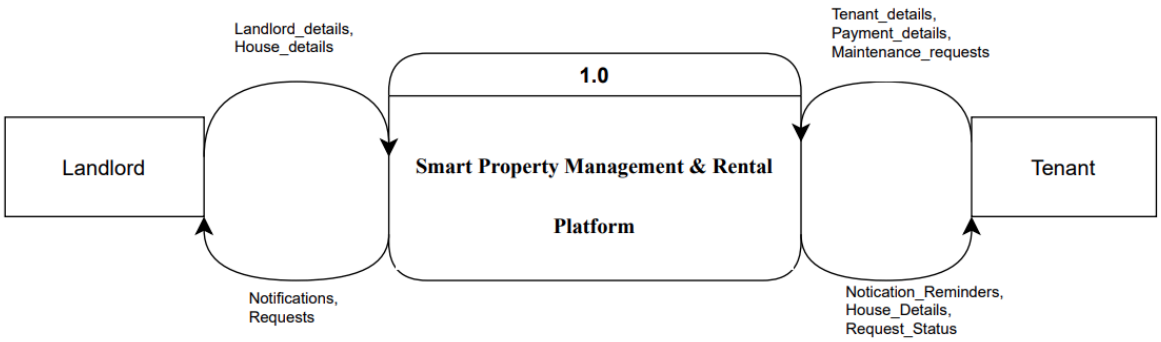

Figure 3: Level-0 Dataflow Diagram

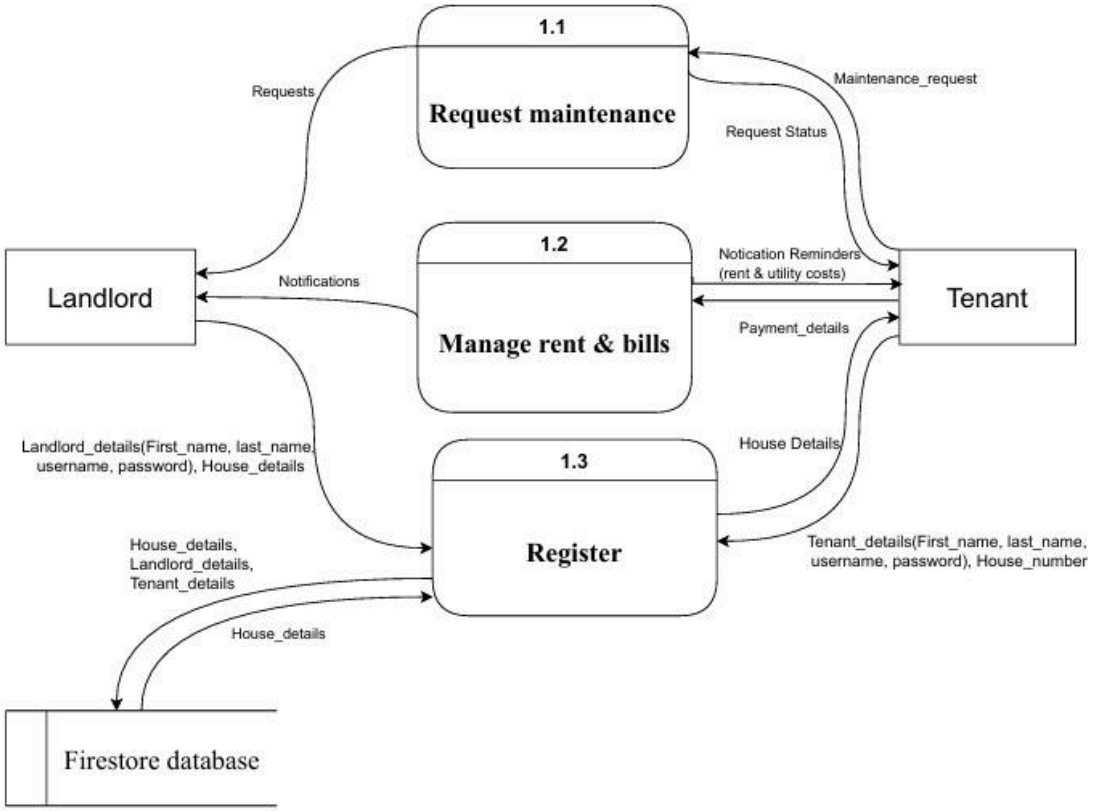

Figure 4: Level-1 Dataflow Diagram

Sequential diagrams

Landlord Interface

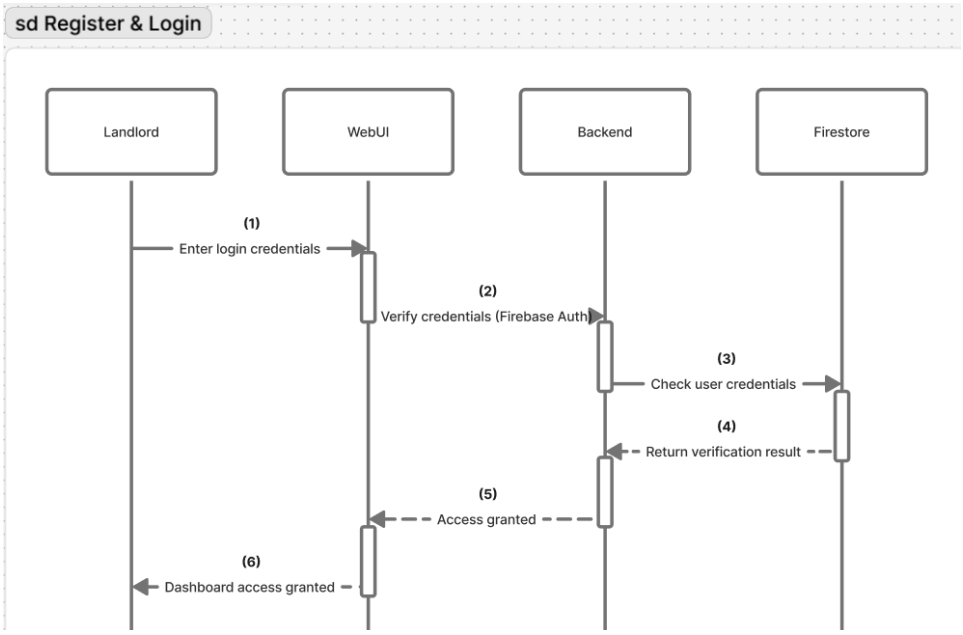

Figure 5: Register & Login

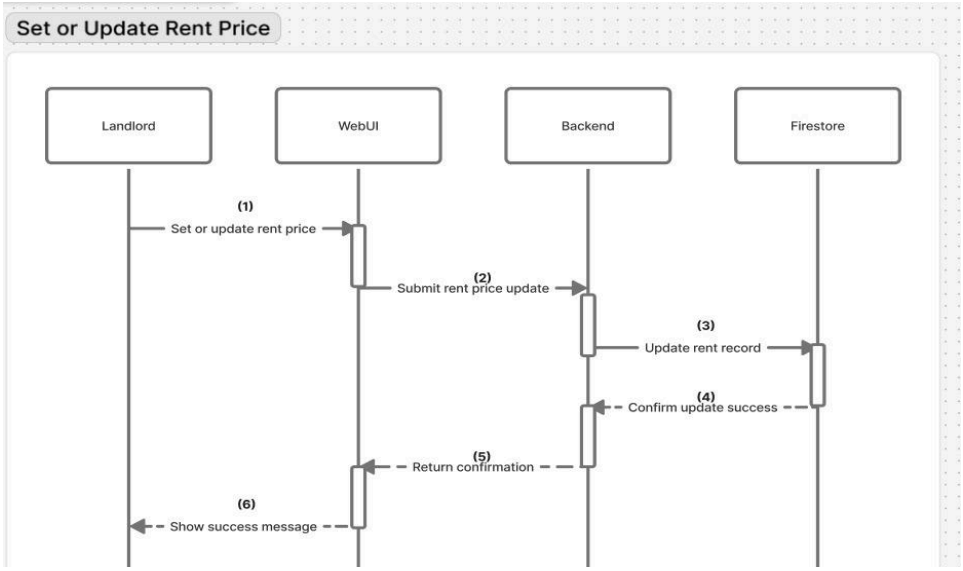

Figure 6: Set or Update Rent Price

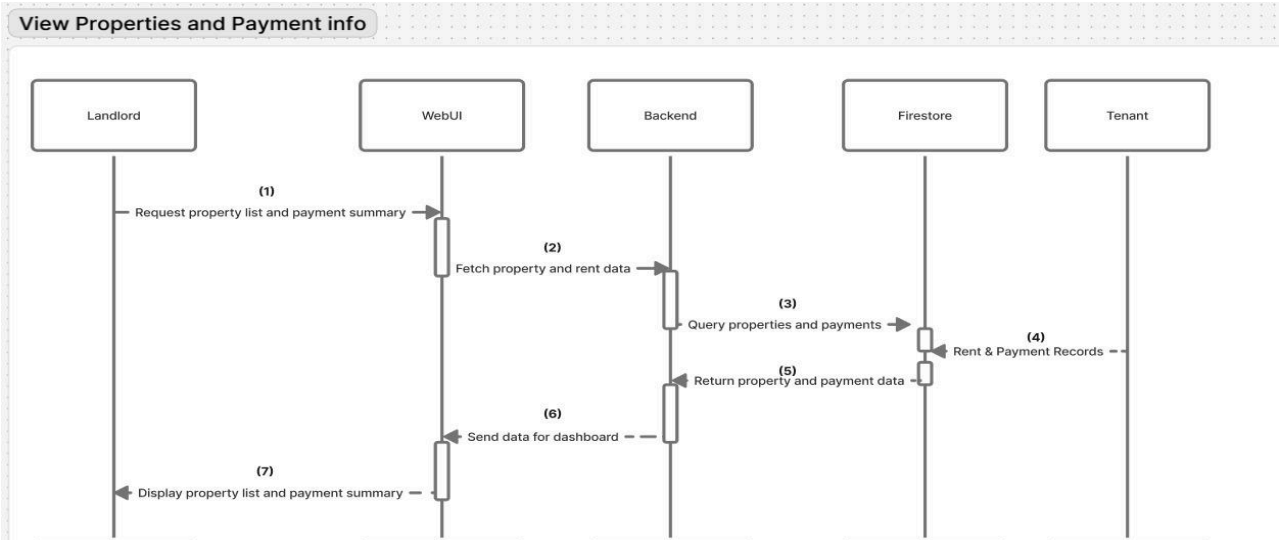

Figure 7: View Properties and Payment info

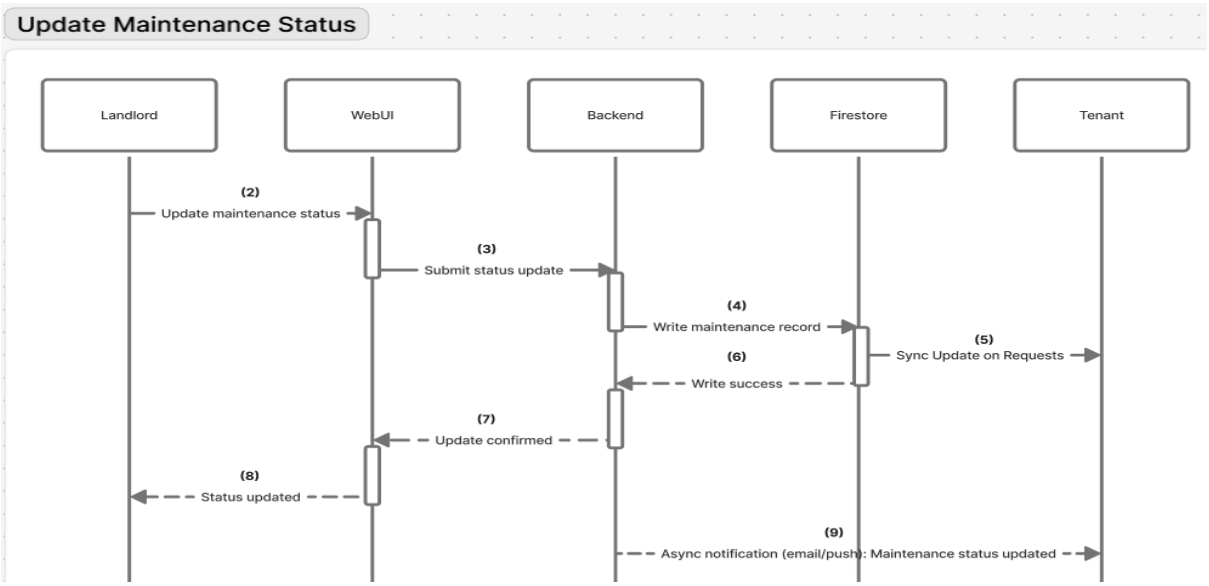

Figure 8: Update Maintenance Status

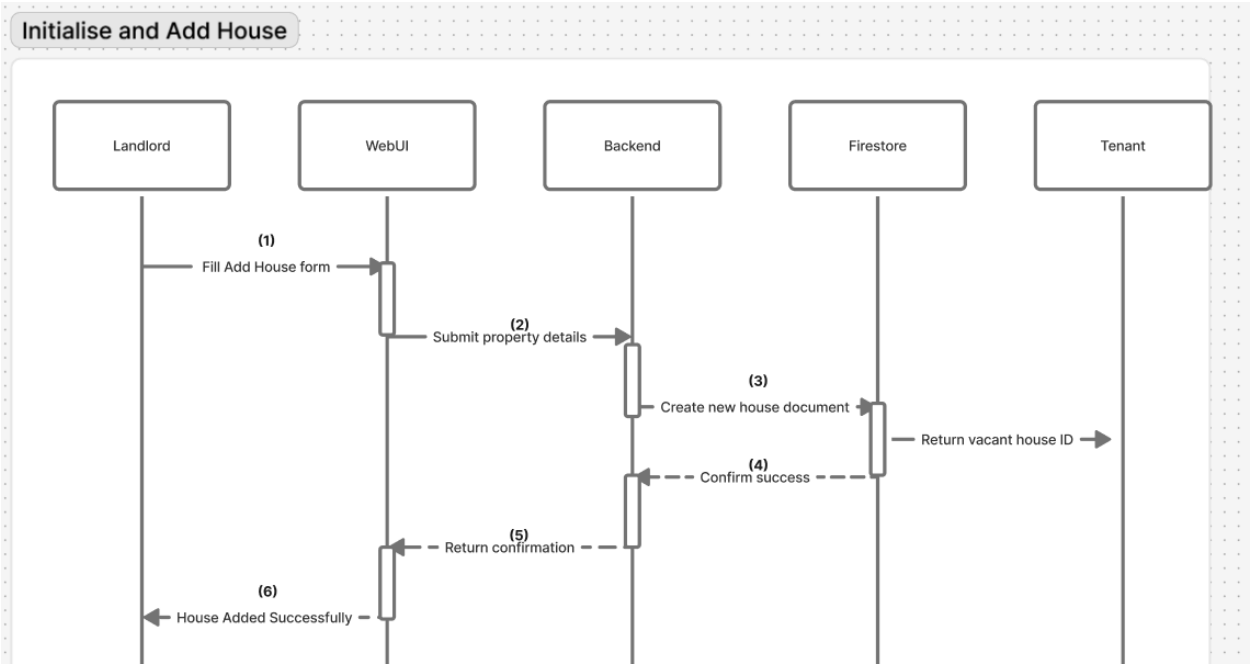

Figure 9: Initialising and Add House functionality

Tenant Interface

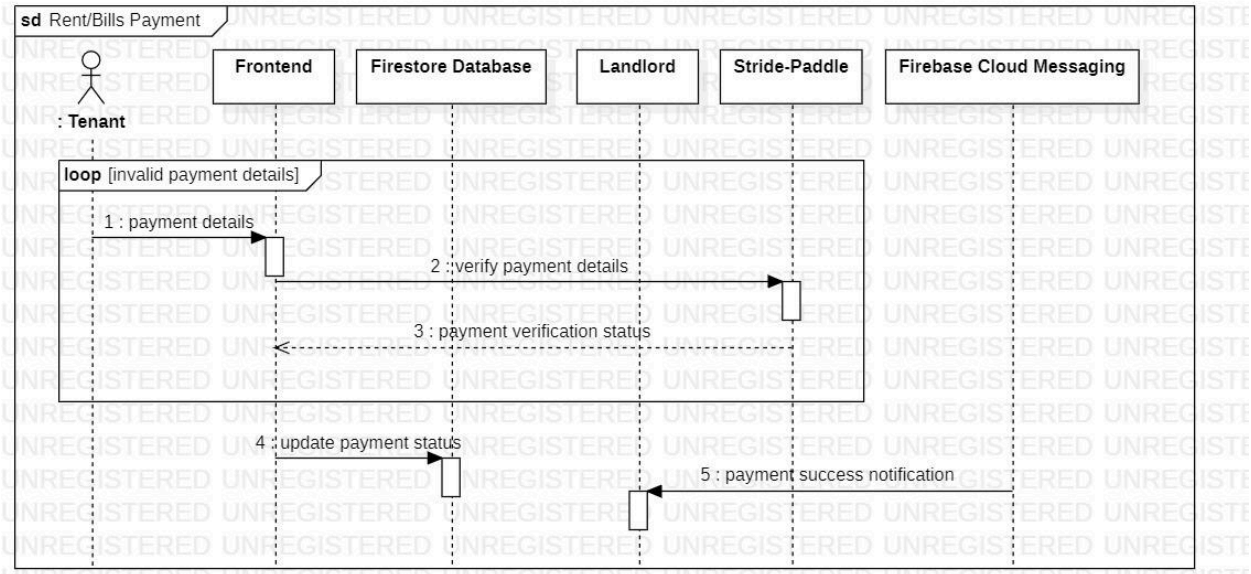

Figure 10: Pay Rent/Bills

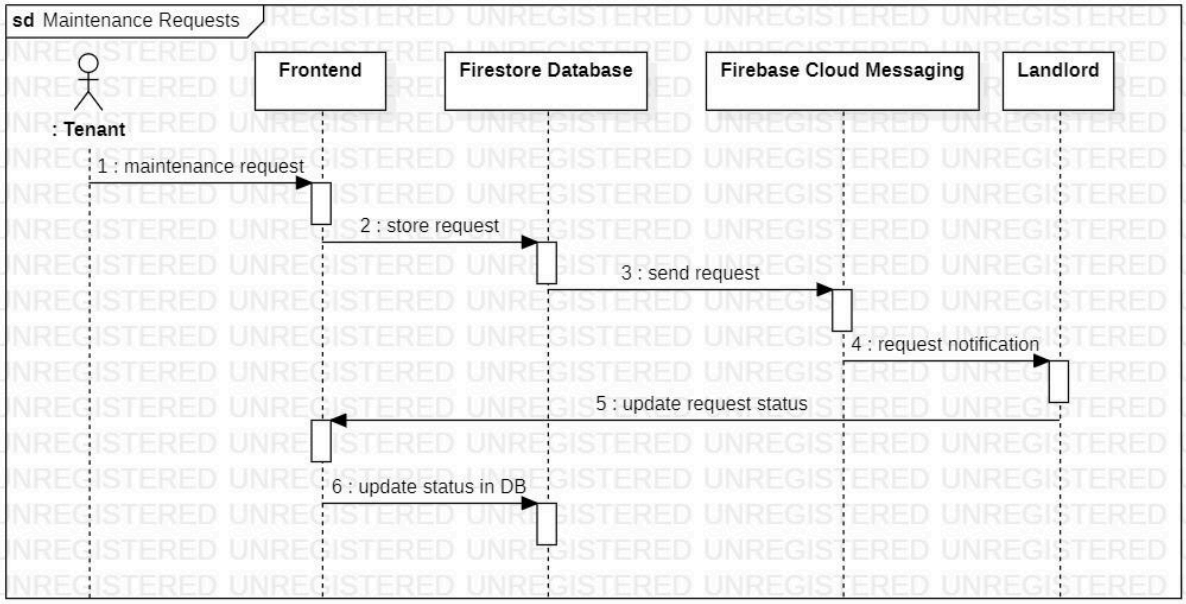

Figure 11: Add Maintenance Requests

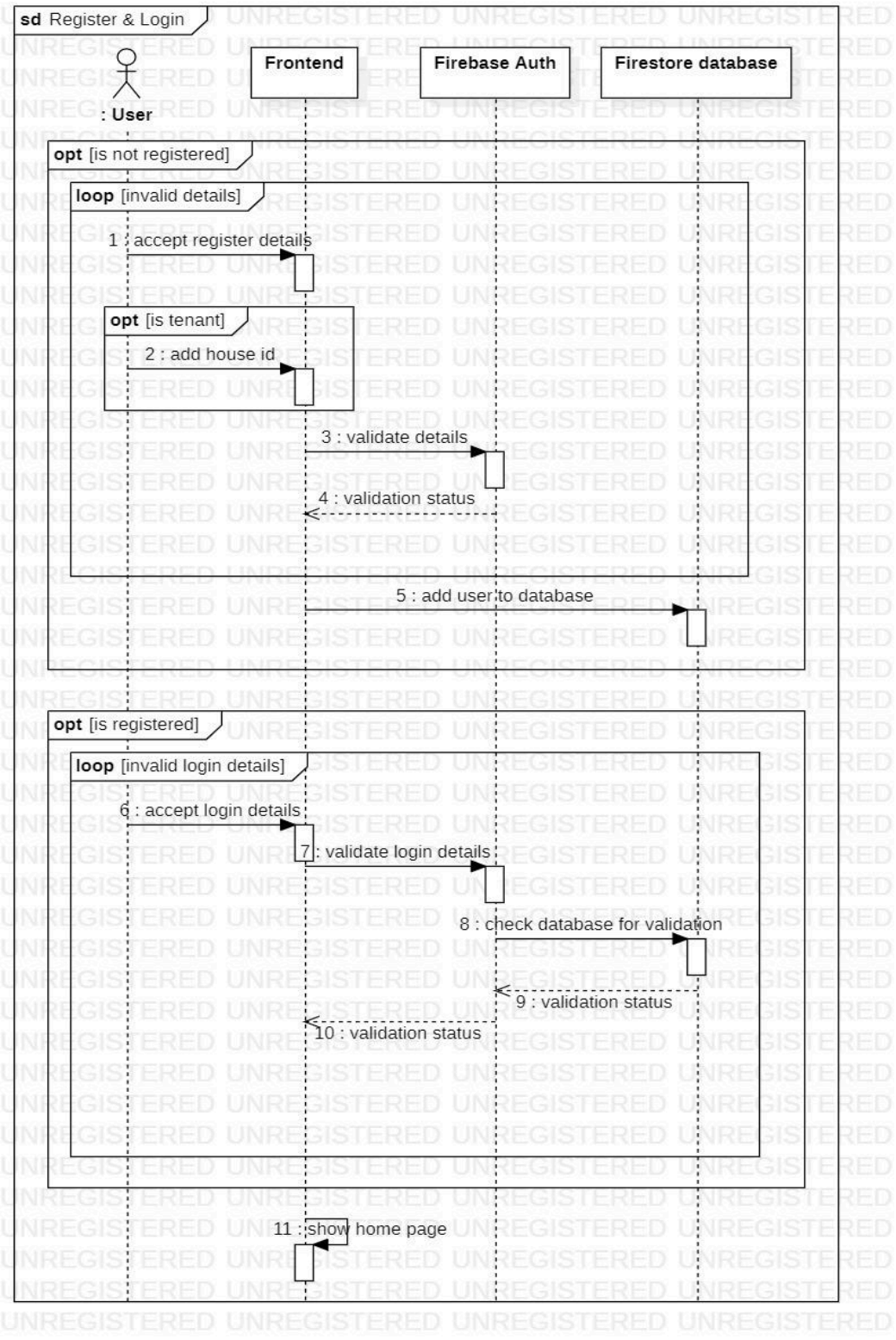

Figure 12: Register & Login

DATA TYPES

- Text (VARCHAR(n)))
- Numbers (INT, FLOAT)
- Boolean
- Date-time/Timestamps
- Metadata

COMPUTATIONS

Authentication:

- Validate credentials or OAuth token.
- Check user type (tenant or landlord).
- Issue a secure session (JWT or Firebase token).

Calculation:

- Rent base + utilities cost calculations

Notification:

- Calculate due dates and trigger reminders
- Decide which users should receive a notification and when based on request forms submission, maintenance status updation and payment verifications

EXPECTED CONTRIBUTION

Frontend designing and development - **Mahlet Bekele**

API development, hosting and integration with frontend - **Zeeshan Imran**

Database setup and hosting - **Miguel Tunga**

System Functionalities and design - **All Members**

Cloud services Integration and hosting - **All Members**

Testing - **All Members**

REFERENCES

1. **Firebase Firestore Documentation** – Official guide to using Cloud Firestore, Google's scalable NoSQL database. https://firebase.google.com/docs/firestore
2. **Firebase Cloud Messaging (Notifications)** – Documentation for integrating push notifications on web and mobile using FCM. https://firebase.google.com/docs/cloud-messaging
3. **Firebase Authentication** *(if you're using login/signup features)* – Securely manage users and roles with Firebase Auth. https://firebase.google.com/docs/auth
4. **Vercel Documentation** – Official documentation for deploying Next.js and other frontend apps with edge functions and CI/CD. https://vercel.com/docs
5. **Render Cloud Hosting Documentation** – Platform guide for deploying full-stack web applications, APIs, and background workers. https://render.com/docs
6. **Stripe API Reference** – Official Stripe documentation for payment integration, billing, and subscription management. stripe.com/docs/api