PROJECT PLAN
Community Sports Facility Management

Team Name: ProjectX_Dev_Hub (Team/Group 7)

Team Members:

- Name: Kamogelo Moetjie - Student Number: 2720816
- Name: Msesenyane Makhongela - Student Number: 2452625
- Name: Thembelani Ngcobo - Student Number: 2548017
- Name: Mpho Kwinda - Student Number: 2549602
- Name: Philani Mkalipi - Student Number: 2567185

Course: Software Design

CourseCode: COMS3009A

Date of Submission: 25 May 2025

Client Name: Martin Shilenge

Contents

1 Project Overview

Purpose / Background

Many communities share sports facilities such as soccer fields, basketball courts, swimming pools, and gyms. These facilities often face challenges related to scheduling, maintenance, and equitable access. Local municipalities or homeowners' associations commonly struggle with coordinating bookings, tracking facility usage, and responding to maintenance issues in a timely manner. This project aims to address these challenges by developing a web-based system that allows users to reserve facilities, report maintenance concerns, and stay updated through event notifications and alerts.

Scope

The project focuses on designing and developing a public-facing web application for managing sports facility reservations and administration. The in-scope features include user authentication using a 3rd party identity provider, facility booking, user role management (Resident, Facility Staff, Admin), maintenance issue tracking, event scheduling, and reporting capabilities.

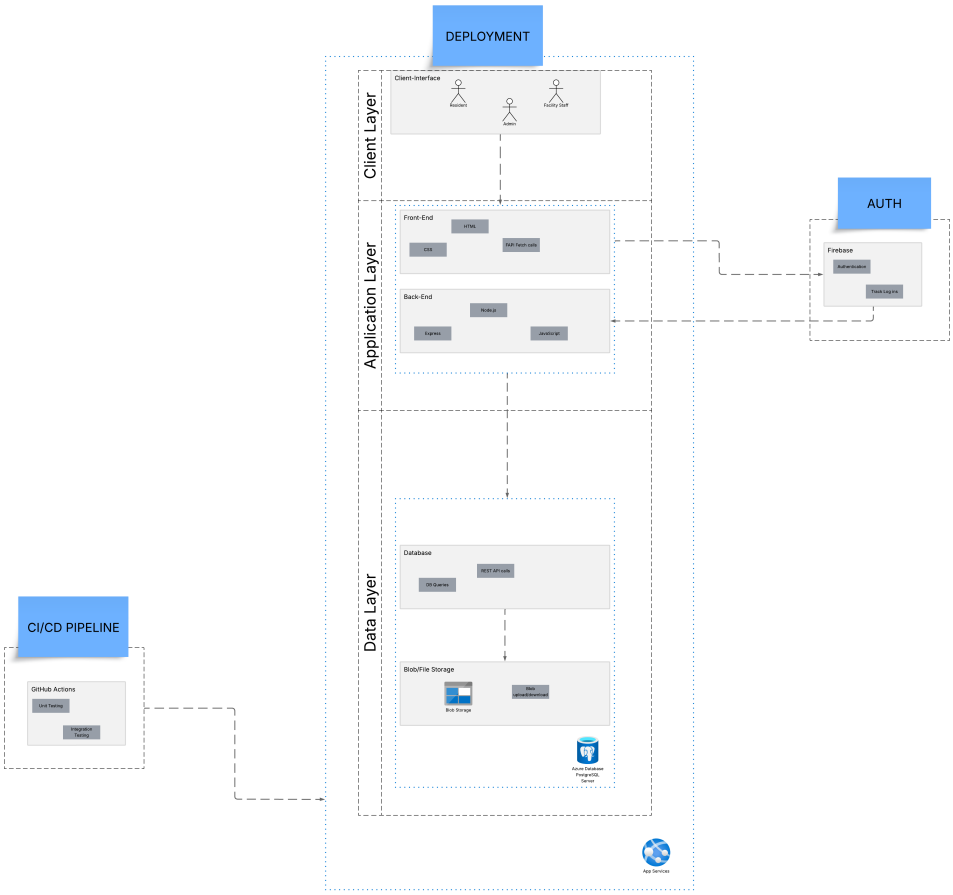
Objectives

The key objectives of this project are:

- Develop a responsive web-based application using an Agile methodology.
- Implement a secure user authentication and authorization system with defined roles.
- Create an intuitive booking system for residents with admin oversight.
- Allow users to report maintenance issues and enable facility staff to track and resolve them.
- Enable event creation and user notifications for upcoming activities or closures.
- Generate at least three exportable dashboard reports for usage trends, maintenance status, and custom views.
- Incorporate CI/CD practices and a test-driven development approach throughout the project lifecycle.

2 System Architecture & Design

High-Level Architecture



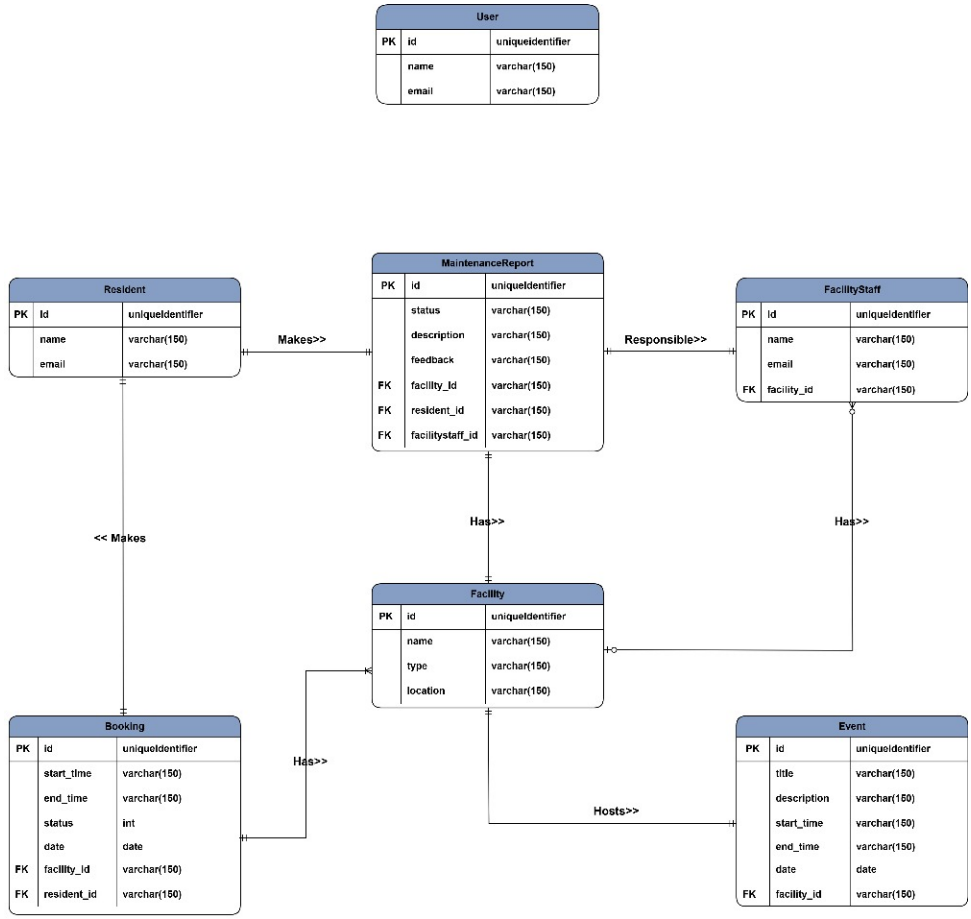
Component Breakdown

- Front-end: HTML and CSS.
- Back-end: JavaScript, Express and Node.js.
- Database: postgres
- APIs: REST, etc.

Technologies Used

Node.js, Firebase (for authentication), Express (API), Postgres SQL (for database), Azure (for deployment and hosting), etc.

Data Models / ERD Diagrams



3 Team Roles & Responsibilities

Name	Role	Key Responsibilities
Kamogelo	Front-end and Backend for Admin pages	Implement the pages for the Admin role f managing users, bookings and viewing analy ics over all facilities.
Philani	Log in, notifications & Auth	Role access & auth, notifications for differe roles.
Msesenyane	Database, Testing & Deployment	Create database tables for bookings, report and events, as well as notifications.
Mpho	Frontend and Backend for Resident pages	Implement the pages for the Resident role make reservations, view upcoming events, a make maintenance reports of the facility.

Thembelani	Frontend and Backend for Facility Staff pages	Implement the pages for the Facility Staff ro to manage facility maintenance reports.

4 Project Timeline

Week	Sprint	Activities	Deliverables
Week 1	Sprint 1	Registration page, login and authentication setup, initial database schema	Auth module, registration & login pages, DB ERD design
Week 2	Sprint 2	Admin page, resident dashboard UI, initial booking logic	Admin dashboard, resident UI, booking back-end
Week 3 & 4	Sprint 3	Booking page, calendar UI, event management interface	Booking front-end, event creation tools, calendar view
Week 5 & 6	Sprint 4	Notification service, analytics dashboard, final integration testing	Notification module, analytics reports, test results
Week 7	Final Submission	Project polishing, Documentations and video presentation	Video presentation on Deployed site, GitHub repository, SCRUM documentations (Backlog, standups, etc) and additional Artifacts (UML diagrams, Testing,etc)

5 Sprints and Backlog Overview

- Project runs in 1- to 2-week sprints (4 sprints total).
- Each sprint begins with planning and ends with review and retrospective.
- The Backlog is tracked using Notion, categorized by roles and modules.
- Burndown charts are updated at the end of each sprint.
- Task tracking link: `https://brick-adapter-2bc.notion.site/1ce54c6a108a8046a088cbe223c9aed0?v=1ce54c6a108a8018b578000c1001cd26`

6 Risk Analysis

Risk	Impact	Likelihood	Mitigation
Member unavailable during sprint	High	Medium	Redistribute tasks early during sprint planning; pair programming for redundancy
Database schema mismatches module changes	High	High	Use schema version control and regular syncing meetings
Late feature delivery due to underestimated effort	Medium	High	Buffer time included in timeline and agile task resizing

Difficulty with CI/CD deployment on Azure	Medium	Medium	Assign a dedicated member to handle DevOps tasks with early testing
Low user engagement with UI	Medium	Low	Conduct early informal usability testing with peers

7 Testing Strategy

- **Unit Testing:** Functions like booking logic, role access, notification triggers.
- **Integration Testing:** Test interaction between backend routes, front-end pages, and database.
- **End-to-End Testing:** Simulate full user flows across roles (Resident, Admin, Staff).
- **Tools:** Postman (API testing), Jest (JS unit testing), and manual test cases via UI walk-throughs.
- **Test Plan:**
  - Write test cases before feature implementation (TDD).
  - Track test coverage for backend and frontend modules.
  - Use automated test scripts for recurring validations (e.g., booking overlaps, login redirects).

8 Environment Setup Requirements

Environment Configuration File (.env)

To successfully run the application locally or deploy it, a '.env' file must be created at the root of the project. This file stores sensitive configuration values such as database credentials and cloud storage connection strings, and is required to start the server or run backend services.

Required Variables

Below is a list of keys and values that should be included in the '.env' file:

- `PG_HOST=sportx-postgres.postgres.database.azure.com`
- `PG_PORT=5432`
- `PG_DATABASE=postgres`
- `PG_USER=postgresdb`
- `PG_PASSWORD=sportX123`
- `AZURE_STORAGE_CONNECTION_STRING= 'DefaultEndpointsProtocol=https;`
  `AccountName=storagesportx;`
  `AccountKey=r4tJcwragYcUC2kznZS+6vCsfxWf7eRo6YYKPA+mfQejDEi/`
  `CJbF7FrQ03wm3ow6xeY/yJ4tl4LC+AStzQUyqQ==;`
  `EndpointSuffix=core.windows.net'`

Notes

- These variables must be set before starting the local development server.
- A sample file (‘.env.example‘) is provided in the repository for reference.
- Do **not** commit the ‘.env‘ file to version control; it is excluded via ‘.gitignore‘.
- In production (Azure), environment variables will be configured securely through the Azure portal.

9 Deployment Plan

- **Hosting Platform:** Azure App Services and Azure Database for PostgreSQL.
- **Deployment Frequency:** End of each sprint and after feature-complete testing.
- **CI/CD Strategy:** Use GitHub Actions to run test suites and deploy to Azure upon push to ‘main‘.
- **Environment Configuration:** ‘.env‘ for secrets, Azure environment variables for production.