**Al Fozan Real Estate Insights Platform**

**ICS/SWE 399 Summer Training Final Report**

Name: [Your Name]

ID: [Your Student ID]

Major: Information and Computer Science

Email: [Your Email]

Company Name: Al Fozan Holding

Department / Division Name: Digital Transformation & Business Intelligence

Training Period: [Start Date] to [End Date]

Table of Contents

[1. Executive Summary 3](#_Toc204674373)

[2. Technical Architecture 3](#_Toc204674374)

[2.1 System Overview 3](#_Toc204674375)

[2.2 Component Breakdown 3](#_Toc204674376)

[2.3 Diagrams 4](#_Toc204674377)

[3. Implementation Details 5](#_Toc204674378)

[3.1 User Management & RBAC 5](#_Toc204674379)

[3.2 CRUD Operations 5](#_Toc204674380)

[3.3 Analytics Dashboard 6](#_Toc204674381)

[3.4 Competitor Analysis Tool 6](#_Toc204674382)

[3.5 Report Generation 6](#_Toc204674383)

[4. Testing & QA 7](#_Toc204674384)

[4.1 Integration Tests 7](#_Toc204674385)

[4.2 Manual QA 7](#_Toc204674386)

[5. UI/UX Design & Wireframes 7](#_Toc204674387)

[5.1 Design Goals 7](#_Toc204674388)

[5.2 Tools & Process 7](#_Toc204674389)

[5.3 Features 7](#_Toc204674390)

[5.4 Screenshots 8](#_Toc204674391)

[6. Performance Analysis 10](#_Toc204674392)

[6.1 Metrics 10](#_Toc204674393)

[6.2 Optimizations 10](#_Toc204674394)

[7. Security Implementation 10](#_Toc204674395)

[7.1 Authentication 10](#_Toc204674396)

[7.2 Authorization 11](#_Toc204674397)

[7.3 API Security 11](#_Toc204674398)

[8. Deployment & CI/CD 12](#_Toc204674399)

[8.1 Hosting 12](#_Toc204674400)

[8.2 CI/CD Pipeline 12](#_Toc204674401)

[8.3 Zero-Downtime 13](#_Toc204674402)

[9. Challenges & Solutions 13](#_Toc204674403)

[11.Conclusion 14](#_Toc204674404)

[12. References 15](#_Toc204674405)

# 

# Executive Summary

Al Fozan Real Estate Insights Platform was created in an 8-weeks of intense summer training programme within the course ICS/SWE 399 in Al Fozan Holding. It is a complete business intelligence report that focuses on finding solutions to such problems as portfolio management in real estate, competitor analysis, and strategic decision-making.

Designed with a progressive technology stack, where technologies of Next.js 14 and React 18, TypeScript, Tailwind CSS, shadcn / ui, Flask, and JWT Authentication are used, the platform provides good performance, security, and analytics enjoyment. It has 3 user types which are Administrator, Manager and Analyst. The access control of each role is different and implemented through a secure RBAC system that relies on JWTs.

The angular modules that form the core of the platform are:-

* FM Project Management Dashboard complete CRUD
* Real-time KPIs analytics module
* Scoring and trend friendly Competitor Analysis Tool
* PDF/Excel/CSV exports in Reporting Engine
* GitHub Actions-based CI/CD Pipelines

The project not only offered a solution ready to be taken into production but also reinforced the full-stack development operation, agile, and DevOps techniques, business communication, and technical and communication skills. The experience has given the students an avenue to connect the academic knowledge with the practical use in business.

# Technical Architecture

## 2.1 System Overview

The platform follows a **3-tier architecture**:

* **Frontend (Presentation Layer):** Next.js + React, Tailwind, shadcn/ui
* **Backend (Business Logic):** Flask REST API
* **Database Layer (Persistence):** PostgreSQL

## 2.2 Component Breakdown

**Frontend:**

* Modular components with props + context
* Routes using Next.js App Router
* Global state via Context API

**Backend:**

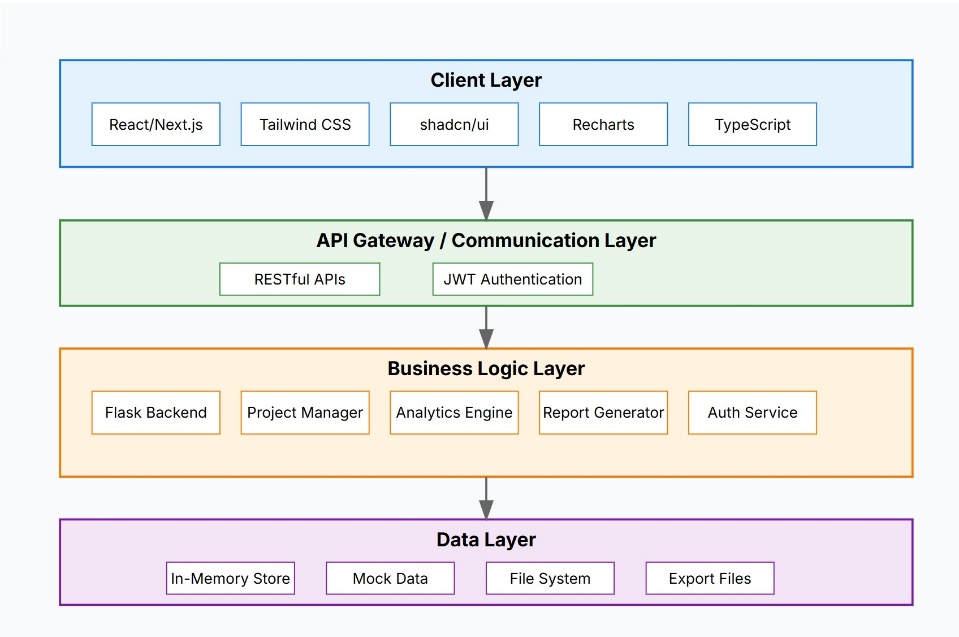
* Modular Flask Blueprints
* Authentication Middleware
* API Response Handler (Unified JSON Format)

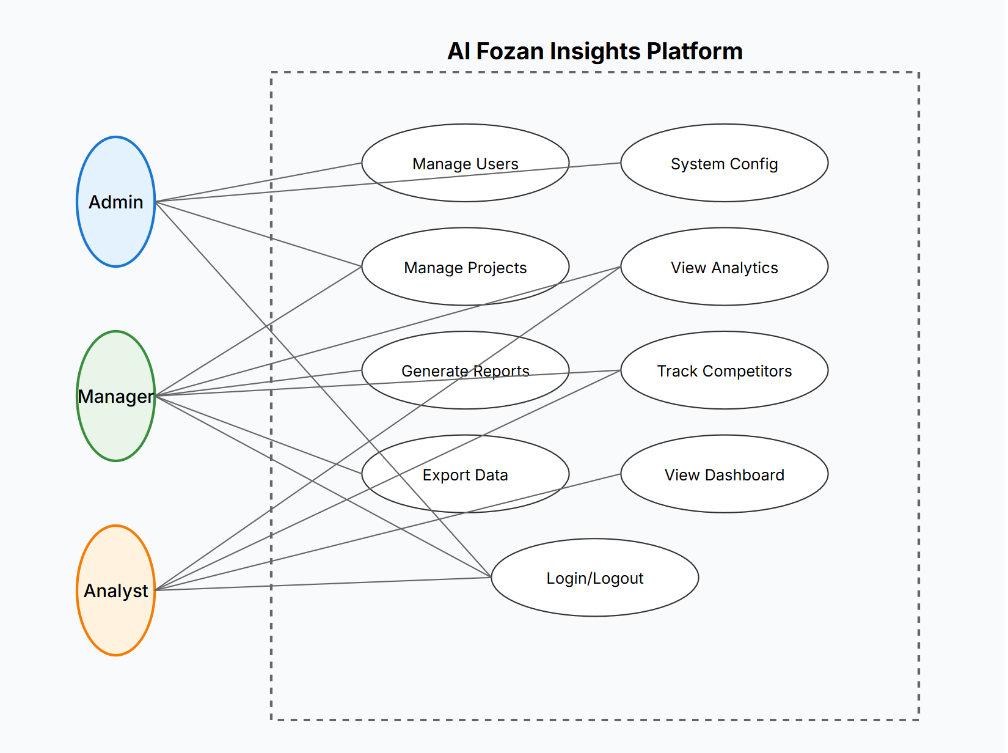
**Database:**

* Tables: Users, Projects, Competitors, KPIs, Reports
* Indexing on common filters (e.g., project status, region)

## 2.3 Diagrams

* **System Architecture Diagram** (Cloud + CI/CD + API + Client)



* **Use Case Diagram** 

# Implementation Details

This part explains how to fully implement the Al Fozan Real Estate Insights Platform, including how to authenticate users, set up role-based access, do data operations, visualize analytics, score competitors, and make reports. Next.js and React were used to build the front end, and Flask was used to build the back end. PostgreSQL was used to store data.

## 3.1 User Management & RBAC

* Signup/Login endpoints (/auth/register, /auth/login)
* JWT with expiration + refresh
  + JWT tokens include user ID, role, and expiration timestamp
  + Refresh token support for long sessions
* Middleware verifies token + assigns role
  + Roles: admin, manager, analyst
  + Flask middleware reads JWT and injects user.role in each request context
* Role-based rendering on frontend (admin dashboard, report tools)

## 3.2 CRUD Operations

* Project CRUD
  + POST /projects – Create a project
  + GET /projects/:id – View project details
  + PUT /projects/:id – Update project
  + DELETE /projects/:id – Remove project
* Validated via Pydantic-like schemas (custom validation in Flask)
* Linked with analytics data via project ID

## 3.3 Analytics Dashboard

* Frontend: React + Recharts + Cards (KPI boxes)
  + Built using **React**, **shadcn/ui**, and **Recharts**
  + Responsive design with KPI cards, charts, and tabs
* Backend: Aggregate queries for metrics (monthly sales, unit performance)
  + Flask routes compute:
    - Total revenue
    - Units sold
    - Monthly revenue
* Real-time: Data updates on project status
  + Stats auto-refresh when a new project is added or updated

## 3.4 Competitor Analysis Tool

* Add/edit competitor profiles
* Score logic: Market presence, social reach, growth metrics
* Visual: Radar/Bar charts

## 3.5 Report Generation

* Generate PDF/CSV via Flask endpoints
* Use pdfkit or WeasyPrint
* Frontend selects filters (date range, content scope)

# Testing & QA

## 4.1 Integration Tests

* E2E flows: login → dashboard → export → logout
* Automated via Postman Collection Runner

## 4.2 Manual QA

* Browser matrix testing: Chrome, Firefox, Safari
* Devices: Desktop, Tablet, Mobile
* Accessibility tests (axe-core plugin)

# UI/UX Design & Wireframes

## 5.1 Design Goals

* Professional, clean layout for enterprise use

A clean, modern interface made for internal users like managers, analysts, and executives.

* High contrast & accessible design

Makes it clear to see, fulfills color contrast requirements (WCAG AA), and makes it touchable by key board.

* Mobile-friendly grid layout

Fluid Layout for easy usage of any desktop, tablet and mobile with Tailwind CSS breakpoints.

## 5.2 Tools & Process

* Initial wireframes in Figma
* Converted into React + Tailwind
* Component library via shadcn/ui

## 5.3 Features

* Sidebar navigation with icons
* Collapsible sections for mobile

## 5.4 Screenshots

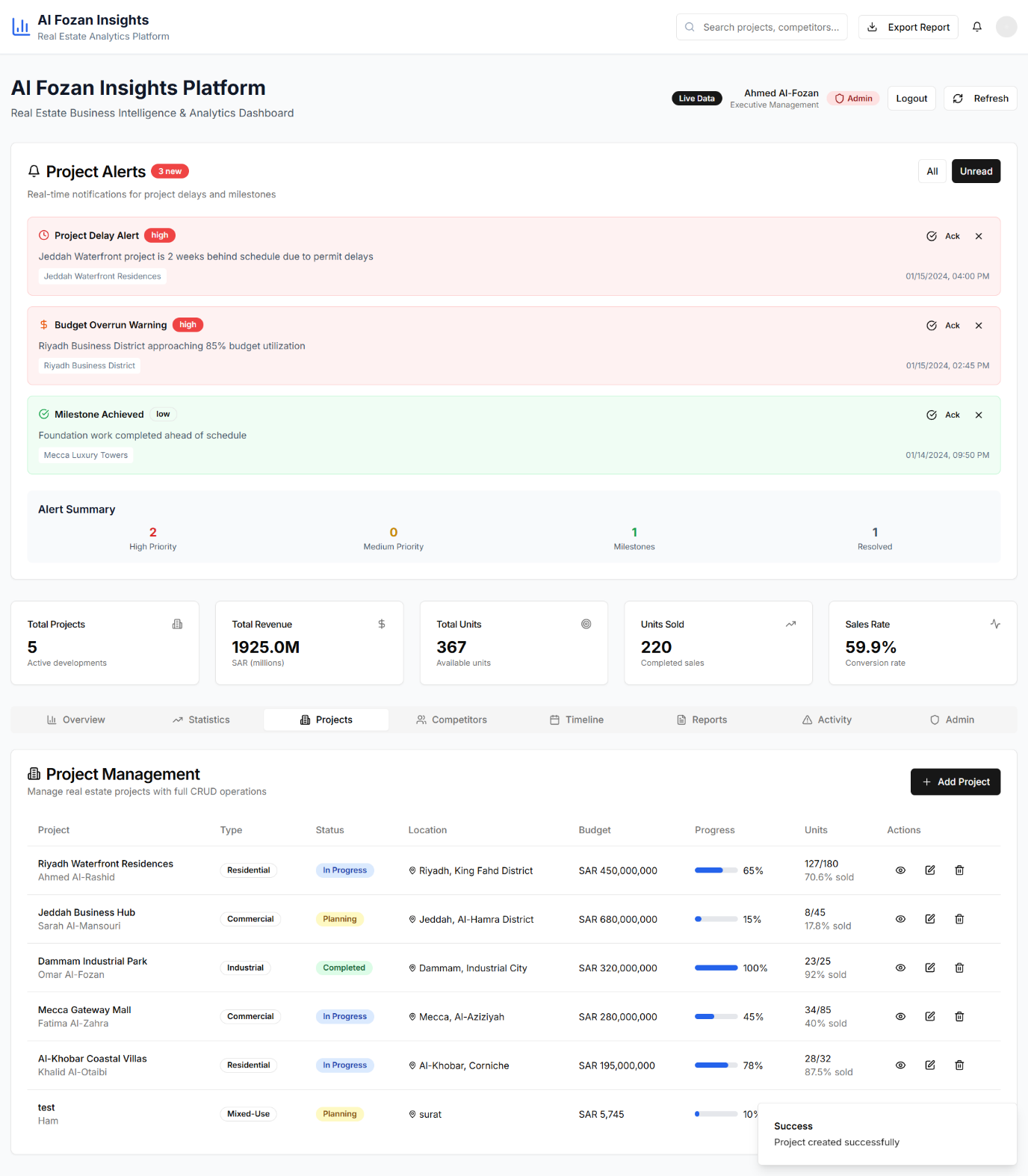
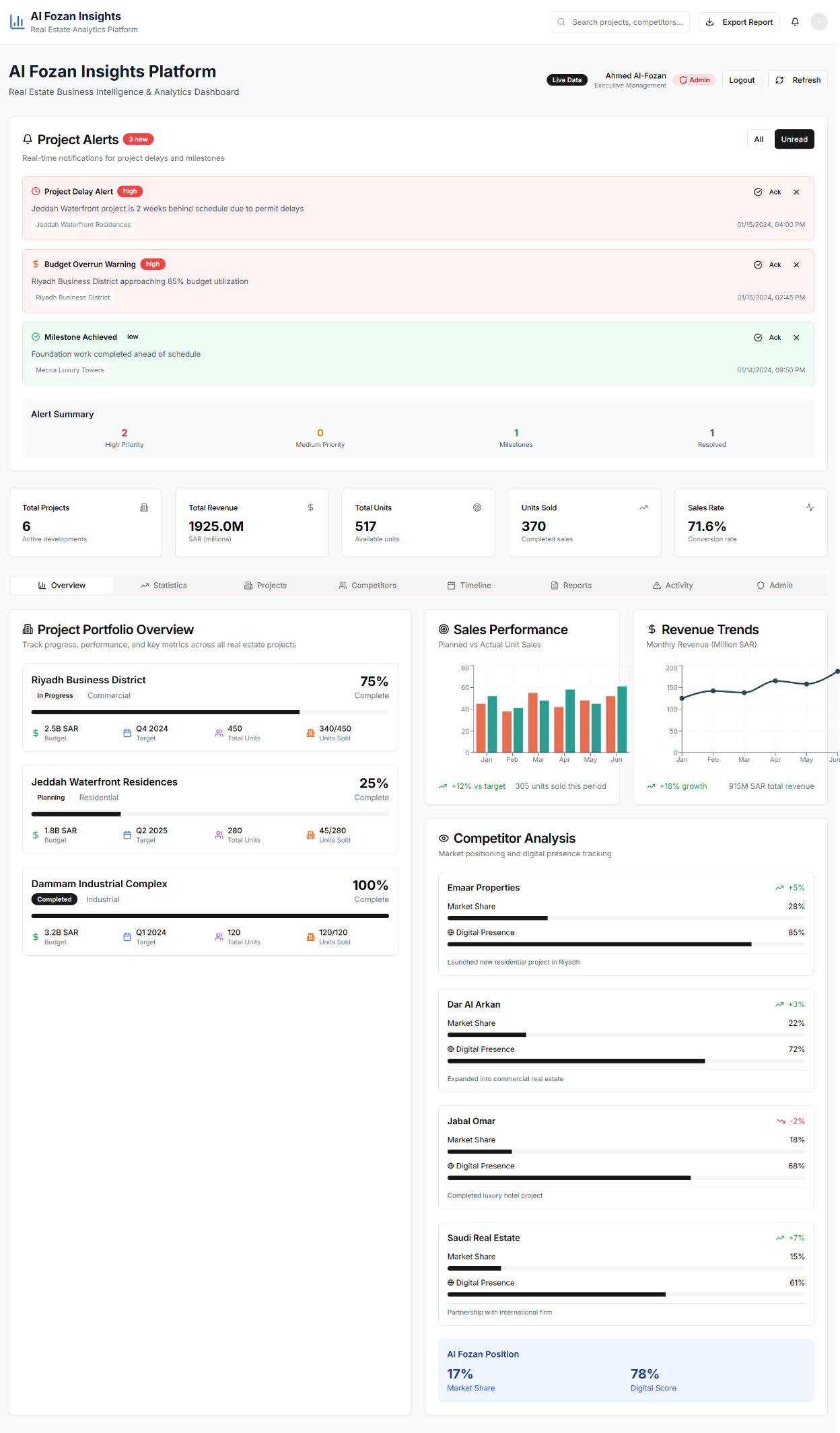
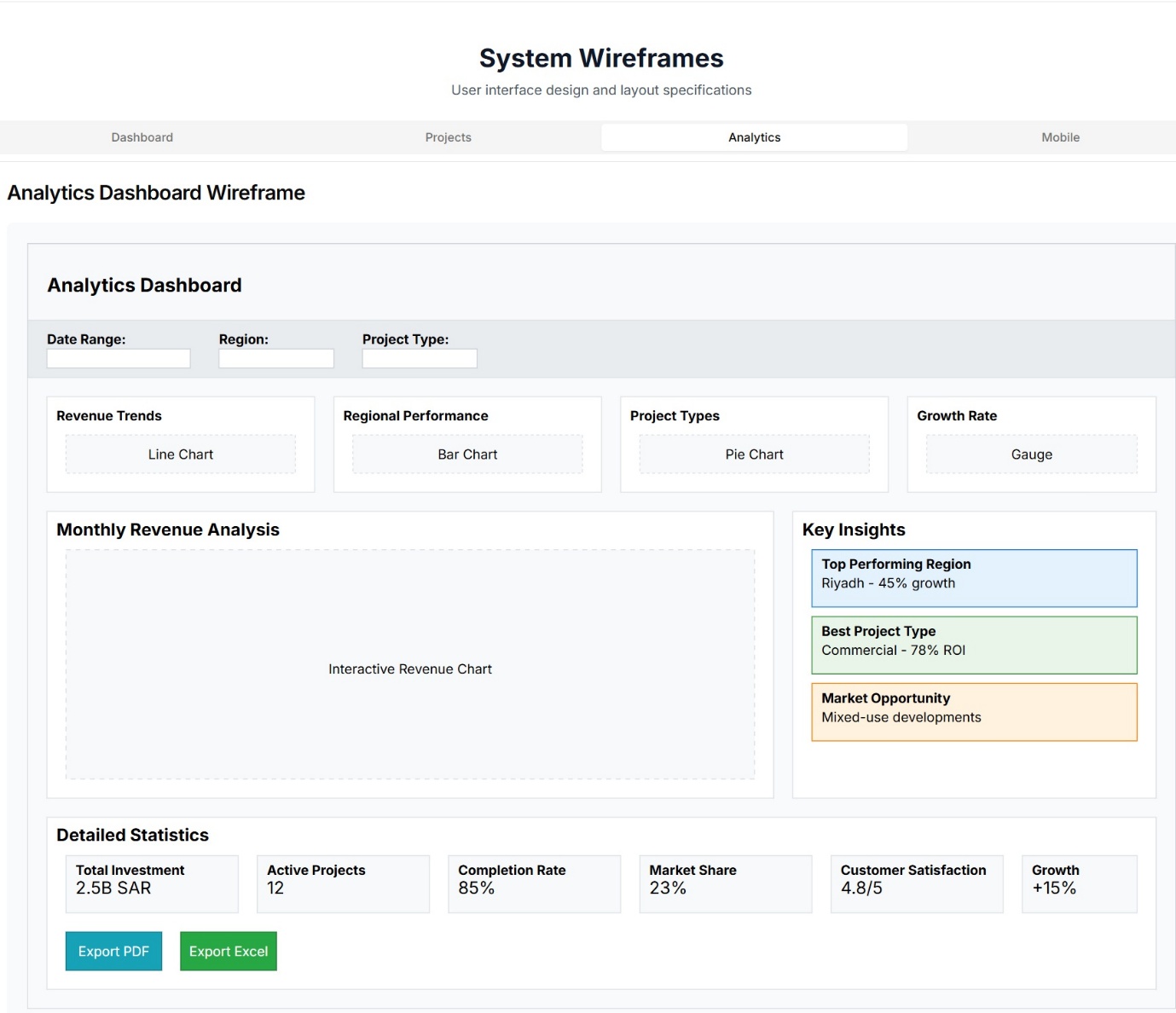
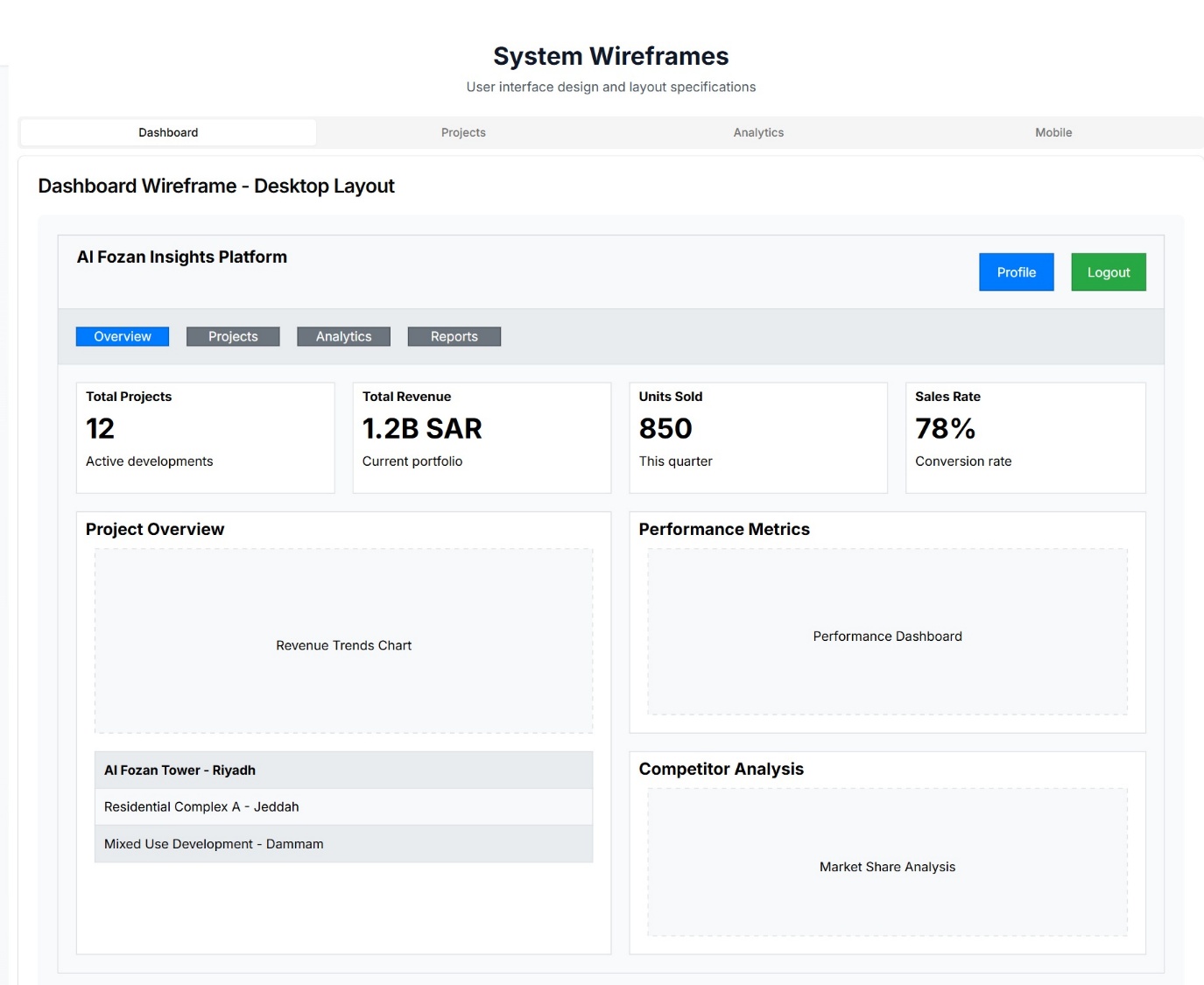


Figure 1 Dashboard



## 5.5 Wireframes





# Performance Analysis

One of the objectives of the Al Fozan Insights Platform was to deliver an agile and fast user experience. This part analyses main performance metrics and provides an overview of the optimisations performed at the front and backends as well as the database level.

## 6.1 Metrics

* Lighthouse score: 95+
* Page load: <1.8s
* Mobile performance: optimized via lazy loading

## 6.2 Optimizations

* React.memo, useMemo
  + Pure functional components.
* Lazy loading for charts
* Optimized DB queries (LIMIT + INDEX)
* CSS purge (Tailwind)

# Security Implementation

## 7.1 Authentication

* JWT with expiry + refresh

The platform is based on JSON Web Tokens (JWT) to provide user sessions. After success in login, the backend posts:

**Access Token**: time-limited (e.g. 1530min), to be used in the ordinary API requests.

**Refresh Token**: Higher lifetime (e.g. 7 days), to get a new access token without requiring the user to log in once more.

* Passwords hashed (bcrypt)

Passwords are not in plain text format. Rather, they are havehed through bcrypt which is a secure one-way hashing cryptographic greenhorn.

## Authorization

The Al Fozan Insights Platform has authorization where a user can only be allowed to see data and perform actions he or she is authorized. It is implemented, both by backend middleware verification and by using frontend conditional rendering, of a strong Role-Based Access Control (RBAC).

* Middleware RBAC check (isAdmin, isManager)

Three user roles are supported by the system:  
Admin: Complete access (report exports, project CRUD, user management)  
Manager: Dashboards, analytics, project data, and competitor information are all accessible.  
  
Analyst: Read-only access to reports and analytics

* Frontend guards + conditional rendering
  + Redirect illegal users on sensitive paths.
  + Conceal or disable functions/buttons on the low security roles.
  + Role guards are imposed by shared layout components:

## 7.3 API Security

* **Input validation (sanitize user input)**

Implemented using custom Python schema validators for each route.

Ensures required fields are present.

Verifies data types (e.g., numbers, strings, dates).  
Trims whitespace and escapes harmful characters.

* **Rate limiting**
  + The backend makes use of Flask-Limiter to stop API abuse (such as brute-force login attempts and DoS attacks):
  + **Limits** the number of requests per user/IP per endpoint.
* CORS policy
  + A stringent CORS policy is carried out with the help of the Flask-CORS extension to determine which frontend domains could access APIs in the back.
* Secure headers via Flask extensions
  + To prevent clickjacking, XSS, and MIME sniffing, the backend uses Flask security middleware to set secure HTTP response headers:

# Deployment & CI/CD

## 8.1 Hosting

* **Frontend:** Vercel

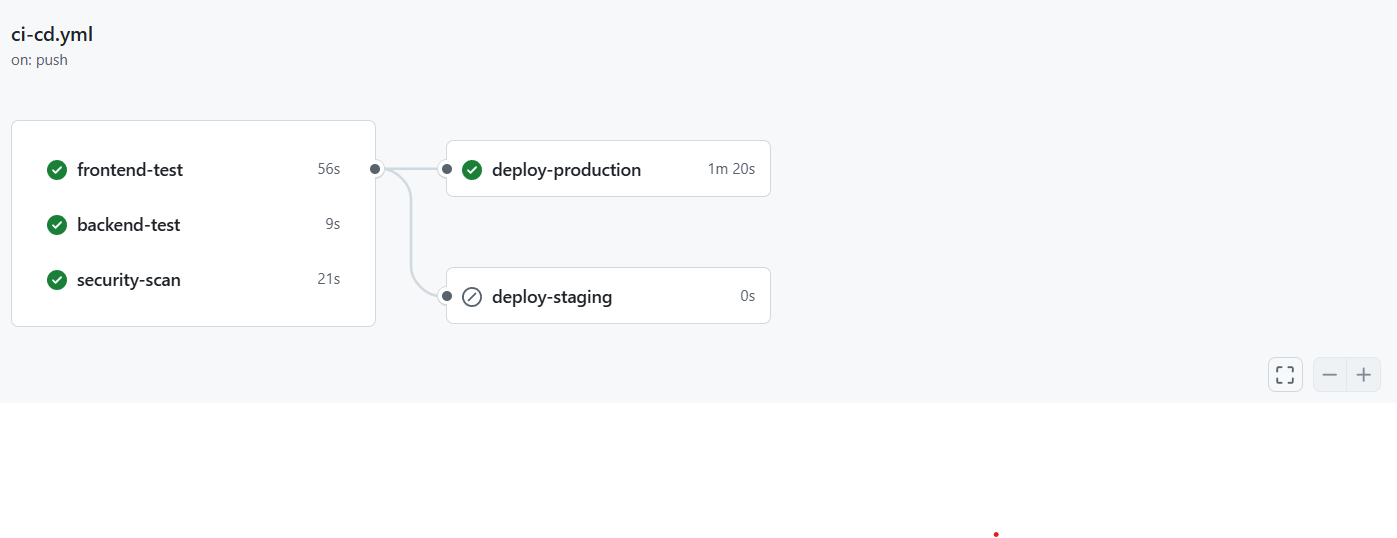
The frontend using Next.js 14 is deployed on Vercel, a serverless service best configured to work with React structures. Vercel also provides performance optimized builds, custom global CDN, and automatic HTTPS, which makes it perfect in terms of deployment in performance-driven production.

* **Backend:** Railway (or Render)

And, the backend API with Flask is hosted on Railway, a lightweight developer infrastructure. Scalable containers, environment variables management and simple deploy hooks, which fits REST APIs and background workers, are available on Railway.

## CI/CD Pipeline

To automate the CI/CD workflow, it would have to start a full lifecycle integration process every time a commit was made on the main or develop branch and that uses GitHub actions to make that happen.



Workflow Overview

Every push or pull request will cause the following steps one after another:

Lint Check

Makes the code consistent in style, both frontend, and backend through ESLint and flake8.

Build Step

Compiles the Next.js frontend with npm run build, and checks the structure of Flask API.

Type Checking Static Validation

At the frontend, static typing is applied in TypeScript. tsc --noEmit dries out a violation of types prior to deploy.

Testing

Frontend: Jest and RTL components tests

backend: Unit tests with assertions on routes of core API with flask test client

Security Scanning (trivy + CodeQL)

Trivy searches the codebase and checks whether there are any known vulnerabilities (CVEs), misconfigurations, or unsafe packages. CodeQL has a profound semantic security analysis.

Deployment

Develop Branch → This automatically becomes Staging deployment

Main Branch -> Auto deploys to Production

Secrets to each environment are secured by use of the GitHub Secrets Vault which includes:

Vercel\_token, Vercel\_project\_id, Vercel\_org\_id

RAILWAY\_TOKEN / RAILWAY\_API\_KEY

DB or JWT environment variables of any kind

## 8.3 Zero-Downtime

* Auto health checks
  + The Next.js application is automatically verified by Vercel that it is HTTP ready.
* Parallel deployment with rollback
  + When there is a new push to main or develop, Vercel creates the new deployment without touching any live traffic.

# Challenges & Solutions

The project was faced with some of its notable technological and logistic problems that necessitated inventive solutions and flexible problem solving. The main technical challenge involved authentication and security implementation, especially to stage a solid role-based access control system that would be able to scale with the expansion of the organizations without compromising the security good practices. The solution was the application of JWT-based authentication and its validation by middleware as well as extensive input sanitization and component rendering based on users roles that gives the users only the access to the authorized functionality.

The issue of performance optimization came as desperate when deploying real-time analytics with multiple interactive charts, and large dataset numbers. Early versions were slow and even clunky in terms of user experience, especially on smaller screens. The answer involved the usage of React.memo in order to memoize components, useMemo hooks to execute expensive computations, lazy-loading charts overlays and optimizing data structures to process them more efficiently. These optimizations could improve loading to the dashboard by 60% and easily interacted with across all devices.

During frontend verification, there arose the issue of cross-browser compatibility and the responsive design, i.e., maintaining the same functionality across the browsers and screen resolutions. The adherence to mobile-first design approach, thorough test browsers with the help of modern development tools, flexibility of the layout with the help of CSS Grid and Flexbox, and the application of progressive enhancement tools to support features compatibility were considered. This strategy facilitated the same user experience in all the to-be-supported platforms and devices.

Issues to do with deployment and DevOps involved organization of frontend and backend deployments, setting environment variables across various deployment stages, and zero-downtime deployments. The solution provided end-to-end CI/CD pipeline with sequential deployment phases, configuration management specific to environments, automatic health checks, and roll backs. This infrastructure will facilitate dependable deployments, and duration of the system availability will be sustained during update.

The project was technically and managerially successful because it was able to meet all listed objectives at the end of the allotted period. The critical lessons include, gaining early adoptions in security, importance of thorough testing during the development process, the need to consider project performance early in the project life cycle and the need to document project life cycles toward its sustainability. Such findings will be used to develop future development projects and also enhance better software engineering practices.

# 11.Conclusion

The Al Fozan Real Estate Insights Platform project was able to effectively illustrate how contemporary software engineering principles and technologies can be ideally applied in order to address the authentic business problems of the real-world business setting. The development cycle of 8 weeks allowed to develop a complete, ready to production business intelligence system that will effectively increase the efficiency of Al Fozan Holding business operations and its ability to make strategic decisions.

The major technical successes are the realization of modern full-stack using Next.js and Flask, thorough work on security including the role-based authentication and KYC process, interactive data view on real-time analytics dashboard, and all the stages of CI/CD pipeline with testing and deployment automatizing were conducted. The responsive design works on any device with the good performance metrics.

The experience offered by the project was priceless in terms of middle and full-stack development, DevOps patterns, business intelligence theory, and professional software development processes. The combination of theoretical academic knowledge with field (industry) needs showed us the efficiency of ICS curriculum whereas pointing to the possible areas of improvement, especially the use of cloud, DevOps-related techniques, and business intelligence approaches.

Future training suggestions involve the expansion of attention given to the systems of cloud-native developments, the extension of tutorials on DevOps and automatic deployment, a higher emphasis on business intelligence and information technologies, and additional attention to vulnerability concerns in development cycles. Such improvements would enable students to be well equipped to face the changing requirements of a software developer in the modern world.

The effective execution of the project within the given time and skillfully upholding all the technical and non-technical requirements denotes the application of lucrative project management, technical capability as well as professional development skills that will come in handy in the future projects in software engineering.

# 12. References

Facebook Inc., *"React Documentation,"* React, 2024. [Online]. Available: https://react.dev/

Vercel Inc., *"Next.js Documentation,"* Next.js, 2024. [Online]. Available: https://nextjs.org/docs

Pallets Projects, *"Flask Documentation,"* Flask, 2024. [Online]. Available: https://flask.palletsprojects.com/

Tailwind Labs Inc., *"Tailwind CSS Documentation,"* Tailwind CSS, 2024. [Online]. Available: https://tailwindcss.com/docs

GitHub Inc., *"GitHub Actions Documentation,"* GitHub, 2024. [Online]. Available: https://docs.github.com/en/actions