Custom-Built Intranet System

WALEED MUSTAFA
01-135221-084
MUHAMMAD USMAN
01-135221-075

Supervisor: Mr Ubaid Ur Rehman

Bachelor of Science in Computer Science

Department of Computer Science
Bahria University, Islamabad

December, 2025

Certificate

We accept the work contained in the report titled "Custom-Built Intranet System", written by Mr. Waleed Mustafa And Mr. Muhammad Usman as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Mr Ubaid Ur Rehman

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator:

_____

Head of the Department:

_____

Abstract

The Custom-Built Intranet System is an intelligent, centralized and secure internal platform, which is created to facilitate communication, workflow operations, document operation and organizational functions. Fragmentation of information, varying system and structure of documents, and manual processes, slowed the existing organizational processes in terms of accessing files, commission computation, sales tracking, and performance reporting. The solution suggested met these constraints by offering structured repositories, standard ized metadata tagging and support files access through CDN to retrieve them faster and enhance data flow. The system that was constructed with the help of Supabase and Flutter provided cross-platform accessibility, real-time synchronization, data integrity, and high performance. The users of role-based dashboards were Admin, CEO, COO, CFO, Sales Manager, Accountant, and HR who were shown KPIs of leads, revenues, commissions, payroll liabilities, user activity. The system also had secure access that was in the form of Supabase authentication, JWT sessions, RLS policies, TLS encryption, multi-factor authentication, and audit trails. The core modules such as financial management, auto mated payouts, tax calculations, sales pipeline analytics, forecasting as well as document management facilitated correct and compliant as well as streamlined operations. Intranet minimized reliance on third-party tools, increased accuracy of the data, standardization of the processes and scalability to expand in future. In general, the system offered an end-to end digital solution to enhance the efficiency of organizations, increase their transparency, and decision-making by providing real-time insights and role-based, secure automation of organizations

Acknowledgments

In the Name of Allah, the Merciful, the Beneficent. May peace and blessings be upon our Prophet Muhammad, and all of His (S.A.W) companions. We are deeply grateful to all those who have contributed to the completion of this project. We would like to express our heartfelt appreciation to our project supervisor, Mr. Ubaid Ur Rehman, Department of Computer Science, for his unwavering support and guidance throughout this project. His tireless efforts and commitment to excellence have been instrumental in bringing this project to fruition. We would also like to extend my gratitude to our friends and family for their encouragement and support, which helped us to stay focused and motivated throughout the project. Without their support, this project would not have been possible. Once again, We express our sincere thanks and appreciation to everyone who has helped us in some way to complete this project.

MR WALEED MUSTAFA
MR MUHAMMD USMAN
Bahria University,
E-8 Islamabad, Pakistan.

December, 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

DSA	Data Structure and Algorithms
OOP	Object Oriented Programming
PF	Programming Fundamentals
SE	Software Engineering
SQL	Structured Query Language
UNESCO	United Nations Educational, Scientific and Cultural Organization
UNICODE	Unique, Universal, and Uniform Character enCoding
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Project Background/Overview

The Custom-Built Intranet System is a web-based enterprise app created with the help of Flutter as the frontend and Supabase (PostgreSQL) as the backend database and authentication system. With the current digital age, organizations are highly relying on technology to control their internal processes, information and communication networks. Nevertheless, numerous small and medium-sized businesses (SMEs) continue to use disjointed software applications to perform sales, financial, documentation, and management operations. Such fragmentation leads to inefficiency, inconsistency of data and slowness in decision-making.

The Custom-Built Intranet System will solve these problems by offering a centralized, role-based system where various departments and users have the ability to collaborate and work in a secure system. It is an in-house communication and administration platform that will consolidate and unify the functions of sales tracking, financial data management, document sharing, user role management, and KPI analytics into one system. Depending on the user role, every user is allocated a custom dashboard based on their tasks and permissions, such as the Admin, CEO, Accountant, Salesperson etc.

The system offers a high degree of scalability, data security and cost-effectiveness as it combines modern cloud technologies and cross-platform compatibility in addition to increased efficiency in the workflow. Its web and desktop support helps organizations to connect to the platform with any operating system without incurring the overhead costs at the back-end, which makes it a low cost and sustainable business solution.

1.2 Problem Description

The day to day activities in most organizations like dealing with clients, tracing leads, dealing with financial statements, and storing documents are carried with different and unconnected tools or even manual means. This strategy poses a huge problem that limits growth and performance. The problems that were revealed were as follows:

- Absence of centralized system results in redundancy of information and data loss.
- Poor management of document and financial records leads to the chances of human error.
- Poor security control systems lead to unauthorized access to sensitive information.
- Late reporting and lack of real-time information impedes efficient outlook among the executives.
- Various applications cause undue complexity and increased cost of maintenance.

In order to overcome these challenges, the project will be offering a Custom-Built Intranet System which will centralize all key operations on a single safe scheme. The system is real time data synchronized, role-based access control and dynamic reporting friendly.

The system aims at offering the following solutions:

- Single portal of access to organizational data.
- Role-specific and secure operations of various categories of users (Admin, CEO, Accountant, Salesperson).
- Automatic data synchronization of sales, accounting and executive dashboards.
- Ease of uploading documents and controlling the version of the same to maintain integrity of data.
- Interactive sales performance and business growth KPI dashboards.

1.3 Project Objectives

The primary objectives of the Custom-Built Intranet System are outlined below:

- To design and develop a centralized system of intranet that will establish a centralized system which will link all departments of an organization under a single platform that is secure.

- To use Supabase to apply safe user authentication and allow role-based access permissions to data.
- To ensure that each user (Admin, CEO, Accountant, Salesperson) has his or her own dashboard and personalized functionality.
- To create a system of organizing organizational documents and financial data in structured SQL tables and metadata tagging.
- To automatically generate and analyze Key Performance Indicators (KPIs) to be reviewed by the management.
- To have a clear approval process to allow the CEO to fix and approve payouts presented by the accountant.
- To improve the security of the data by using encryption and audit logs, Supabase Row-Level Security (RLS) policies.

1.4 Project Scope

The project is aimed at designing and developing a secure, scalable, and friendly intranet system which makes internal operations and data flow among various functions in an organization to be simple. It is based on desktop and web platforms with the responsive interface of Flutter and Supabase as a free and trusted backend service.

1.4.1 Inclusions:

- Supabase Auth access control and authentication based on roles.
- Admin module that handles user, role and audit log.
- CEO module to track KPIs, financial reports and grant payout requests.
- Invoice, tax and payment records module of an accountant.
- Salesperson module where leads, clients and contact details are added along with the uploading of documents.
- Metadata tagging and access control Secure document management.
- KPI dashboard for real-time performance tracking.
- Audit log to track users and changes in data.

1.4.2 Exclusions:

- None of the integration with third-party ERP or CRM applications.
- No mobile (Android/iOS) application at current stage..
- None of the external payment gateway integration (processes internal records alone).

1.4.3 Target Users:

- **Admin:** Creates and maintains users, assigns roles, logs activities and ensures the security of data..
- **CEO:** Checks KPIs of sales and finance, reviews tax reports and approves or disapproves payout requests.
- **Accountant:** Posts, updates and removes financial materials, invoices and tax statements.
- **Salesperson:** Incorporates leads, clients, and contact information, uploads documents and maintains his or her record.

1.4.4 Project Deliverables:

- An intranet system that is fully functional and controlled by roles.
- Secure authentication on Supabase database with the central database.
- Live sales, financial and management dashboards.
- Documents and reporting management automation.

1.5 Summary of Key Features:

- **Login:** The Supabase authentication is email/ password based. The session management secures the privacy of data.
- **User Role Management:** The admins can give roles (Admin, CEO, Accountant, Salesperson) and give control over each.
- **Document Uploads:** Invoice, report and document uploading allow metadata tagging of the documents so that users can easily retrieve them.
- **Financial Module:** Tax reports, invoices, payouts can be added, and payment requests can be managed by the accountants.

- **KPI Dashboard:** The CEO is able to see the performance of the organization at large and compare the sales and finance data.
- **Audit Logs:** The admins will be able to track activities of the users, data updates, and access patterns to ensure accountability.
- **Lead Management:** Salespersons are able to add and maintain information and sales leads of clients.
- **Approval Workflow:** The Accountant can present payouts and other financial requests to the CEO and approve or disapprove it.
- **Search and Filter:** Every module consists of search and filtering to have a good command of data.

To summarize, the **Custom-Built Intranet System** is a digital workspace that helps to increase employee collaboration and improve the privacy of data turnover, as well as manage business operations in various departments. Through the flexibility of Flutter and the modernity of Supabase back-end services, the system offers an effective, free of charge and scalable platform that can be used by organizations that seek to digitalize their inner processes.

Chapter 2

Literature Review

2.1 Existing Systems

An Intranet System refers to any internal and private network that is used to promote communication, exchange of information and workflow management in an organization. The collaborative features offered in modern intranets include document-storage, internal messaging and dashboards to track major metrics. According to [1], an intranet is a central online workspace, which links employees, management as well as resources in a monitored setting.

Many enterprises have adopted platforms such as Microsoft SharePoint, Google Workspace and Zoho WorkDrive for internal collaboration. Nevertheless, many of these systems may be very expensive to license and may have complicated set up and hence they may not be accessible by small to medium enterprises (SMEs). [2] claims that intranet failures are commonly due to bad governance, no ownership and unmanaged content.

Research by [3] argues that role-based intranets also increase user experience by limiting information overload based on individual dashboards. The content is accessed by each employee concerning their department or role, and this is more useful and increases the adoption rates.

[4] underline that the systems of intranet are significant in making decisions because they are used to centralize corporate information and analytics, facilitating managerial awareness. Moreover, enterprise portals combine various applications, using the one sign-on system, which incorporates HR systems, finance systems, and documentation systems to enhance its performance and transparency[5].

2.2 Role-Based Access Control (RBAC)

Intranet systems have major requirements in security and data confidentiality. One of the well-established mechanisms is the **Role-Based Access Control** (RBAC) model enabling users to use the resources only according to their roles. [6] [7] presented a corporate RBAC model and reference implementation, and the capability to impose a common authorization over distributed intranet architectures. Their use enabled web services to be secured with a centralized set of policies available as opposed to a set of permissions.

[8] proposed a role-based access control model that is specific to intranet systems, isolating both global and local roles to provide access to organizational information flexibly but in a secure manner. The researchers established that RBAC is effective in terms of minimizing administrative overhead and increasing security and auditability. Recent developments in [9] enhanced RBAC with dynamic context-based policies based on UML and Event-B modeling to resolve real-time security requirements.

Other models such as the means of the Attribute-Based Access Control (ABAC) and Relationship-Based Access Control (ReBAC) were introduced in [10]. Nonetheless, RBAC still can be more viable in the context of enterprise systems with definite hierarchy and foreseeable permission groups.

2.3 Flutter and Supabase Integration in Modern Systems

The intranet systems that are being used today have a growing trend of utilizing open-source systems in order to provide cost-effectiveness and scalability. The **Flutter + Supabase** combination (i.e., cross-platform UI, and backend services) has gained popularity with the development of web and desktop applications where synchronization in real-time and secure authentication are needed.

[11] offers official documentation of the interaction of Flutter applications with the backend of Supabase, such as authentication, SQL queries, and files management. [12] talks about Supabase and its integration with Flutter to perform CRUD operations, highlighting that the tool is open-source and thus more affordable than Firebase. Additionally, [13] presents a comprehensive implementation guide on how to bind Flutter apps to Supabase, which can be used to take database operations and securely store data.

The supabase_flutter package, as referenced in [14], makes integration with the client easier, enabling the developer to develop intranet-like applications without much configuration on the backend. These technologies are collectively capable of delivering fast, safe, and scalable application development, thus, it is suitable in a Custom-Built Intranet System.

2.4 Summary

According to the literature reviewed, it is clear that intranet systems have grown in terms of being mere communication tools to become integrated business management tools. Scalability, governance and access control challenges are however still there. Role-based access control systems such as RBAC remain prevalent because they are simpler in structure and offer high security levels. Combining Flutter with Supabase is a new efficient solution to create cost-effective, cloud-friendly intranet systems that are safe and highly personalized. The suggested **Custom- Built Intranet System** will utilize the following technologies to address the gaps that are already in place by integrating document management, KPI analytics, financial processes, and role-based access into a single system.

Chapter 3

Requirement Specifications

3.1 Introduction

Requirements specification and analysis are what the system should be able to do and the quality it should be able to do it. These requirements should be: quantifiable, relevant and detailed. This chapter gives the requirements of the **Custom- Built Intranet System** (CBIS) in two sections functional requirements, non-functional requirements and lastly, use case models. The CBIS uses four main roles **Admin**, **CEO**, **Accountant** and **Salesperson** and will be deployed with the help of **Flutter** (Web/Windows) and the following as the backend which is **Supabase** (PostgreSQL, Auth, Storage).

3.2 Existing system

Organizations use patchwork systems with most leads/ clients, financials, KPI dashboards, files, approvals. This fragmentation brings in a number of limitations:

- **Siloed data and duplication:** The data of sales, finance, and management exists in disparate applications leading to inconsistent records and overhead in the area of reconciliation.
- **Manual approvals and weak traceability:** Payout and finance approvals are often conducted via email or chat, lacking audit trails and version history.
- **Poor role alignment:** Generic tools fail to make specific views to particular positions (Admin/CEO/Accountant/Sales) with resulting information overload and security hazards.
- **Limited searchability:** Documents do not have homogeneous metadata tags thus retrieving them is slow and error prone.

- **Cost and maintenance:** Commercial suites are expensive to license and their administration is complicated and does not fit in SMEs.

3.3 Proposed System

The **Custom-Built Intranet System** standardizes internal functions into one (secure) role based platform. It uses Flutter and Supabase and has:

- **Authentication and RBAC:** Supabase Authenticated role based dashboards (Admin, CEO, Accountant, Salesperson).
- **Admin module:** User/role administration, user activity logs and system measurement (number of users).
- **Sales module:** Contact, lead, and client management and upload of the document.
- **Finance module (Accountant):** Documents (financial (invoices, tax reports), payout and payment request processes), metadata tagging, and edits.
- **Executive module (CEO):** KPI dashboard (sales, payouts, taxes), overview of financial reports, approving/denying payout request.
- **Document management:** Uploads without insecurity to Supabase Storage with metadata, versioning tricks, and access control.
- **Search & filters:** Text search and tag based filtering of modules.
- **Auditability:** System-wide record of activity in terms of inserts/ updates/ deletes and approvals.

3.4 Requirement Specifications

3.4.1 Functional Requirements:

- **User registration and login:** The user creates an account and secures the account through email/password (Supabase Auth). Sessions are kept; flows of password reset are enabled.
- **Role-based dashboards:** When the user logs in, he or she is taken to a role-based home (Admin/CEO/Accountant/Salesperson).
- **Admin: User and Role Management**
  - Add/edit/delete users; allocate roles and account status (active/ inactive).

  - See overall users and statistics of recent activity.
  - Reset user passwords / send password reset mails..

- **Salesperson: Lead/Client/Contact Management**
  - Add/edit/delete Leads, Clients, and Contacts.
  - Post and attach supportive materials (e.g. proposals, contracts).
  - Filter and search according to status, the owner, date of creation and tags.

- **Accountant: Financial Documents & Payouts**
  - Add/search/edit/delete financial documents(invoices, tax reports) with metadata.
  - Make payouts and payment request amounts, beneficiaries, and references.
  - Statuses of the track: Draft, Submitted, Approved, Rejected, Paid.

- **CEO: KPIs & Approvals**
  - Examples of KPIs that should be viewed include sales totals, invoices aging, taxes by period, payout pipeline.
  - Check and approve/decline payout requests commenting on them.
  - Point and drill down to underlying transaction and documents.

- **Document Management**
  - Upload/download files to Supabase Storage with metadata (title, tags, entity links).
  - Role-based access to documents.

- **Search, Filters, and Audit Logs**
  - Search between entities, full-text and tag.
  - CRUD and approvals audit log (who, what, when) on a system-wide basis.

- **General Settings**
  - Update of user profile (name, password).
  - Logout and session termination.

3.4.2 Non-Functional Requirements:

- **User interface:** Flutter (UI) is responsive, contains accessibility and clear navigation issues.

- **Security:** Supabase Row-Level Security (RLS), least-privilege role design, transport security (TLS) and least-privilege role design.
- **Performance:**Nominal load Page load less than 2 seconds and KPI queries less than 3 seconds.
- **Reliability:** Availability SLA 99.9 percent (Supabase managed hosting); backups are enabled on a nightly basis.
- **Scalability:** Supabase infrastructure and CDN of static resources do horizontal scaling; scaling 500+ simultaneous users.
- **Compatibility:** Chrome/Edge on Windows/macOS Latest, Flutter windows desktop running of kiosk-based offline-capable intranet (optional).
- **Audit & Compliance:** The logs of activities are fixed and saved in per data retention policy

3.5 Overview of Actors and High Level Use cases

The figure 3.1 illustrates the overall functional hierarchy of the Custom-Built Intranet System, which depicts the way various user functions relate to the system via the Flutter frontend and that all processes are facilitated by the Supabase backend. All actors, including Admin, Salesperson, Accountant and CEO, have access to the features of their role, which Supabase implements via Supabase Row-Level Security (RLS) and role-based access control.

The system starts with a Login use case that a user is required to carry out to access his or her portals. Authentication will also be done using Supabase Auth, which provides secure access using encryption and role verification. Once the user is logged-in, he or she is redirected to various modules depending on the assigned permissions.

The Admin Module enables the Admin to control the core system settings. Admins are able to control users and roles, inspect activity logs and look at internal metrics. These functions are important to guarantee the administrators the ability to regulate access, recruit new workers, and govern the system in general. Any action of the admins communicates with the Supabase database so that user tables, audit logs and permissions can be updated.

The Salesperson role is that of Sales Module. In this case, users are able to deal with leads, clients and contacts via a dedicated Sales Portal. The Flutter front end generates sales data sent to Supabase tables to allow real-time financial module syncing, forecasting, and managing pipeline.

The Accountant works with the Finance Module. It comprises the capabilities of working with financial documents (invoices, tax reports) and developing payouts and

payment appeals to the staff. Flutter sends and receives financial data to Supabase as secure RPC calls when the accountant updates or submits financial data to determine correct calculations and record integrity.

The CEO is provided with the Executive Module. It also grants access to the review of payout/payment requests, approve or disapprove payouts, and the general company performance by using the KPIs Dashboard. Supabase real-time queries and materialized views will power these dashboards, enabling the CEO to make informed decisions in a short period of time.

In all modules, such common features as Document Upload and Document Search and Filter are provided. They are based on Supabase Storage to manage files and Supabase Database to store metadata and tags allowing rapid access and removing the issue of siloed and unformatted documents.

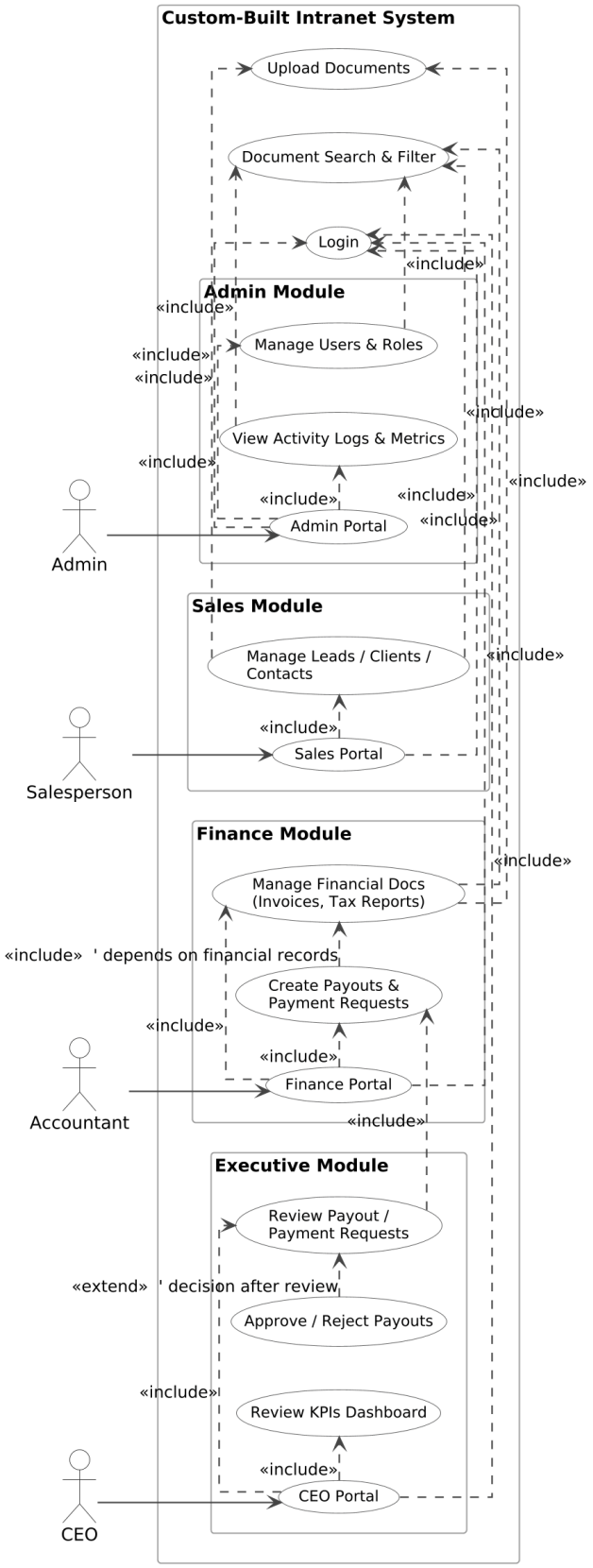

Figure 3.1: Custom-Built Intranet System – Overview of Actors and High-Level Use Cases

3.6 Use Cases

3.6.1 Use Case Tabular for User Login

Table 3.1: User Login

Use Case ID	0001
Use Case Name	Login
Actors	Any User (Admin/CEO/Accountant/Salesperson)
Description	Authenticate user and route to role-based dashboard
Trigger	User clicks “Login” and submits credentials
Pre-Conditions	User has a registered account and valid credentials
Steps	1) Enter email/password 2) Submit 3) System verifies credentials 4) Redirect to role dashboard
Post-Conditions	Session established; access per assigned role
Special Requirements	TLS, JWT session, RLS enforced
Assumptions	Network connectivity and Auth service available
Comments	Supports password reset if needed

3.6.2 Use Case Tabular of Signup

Table 3.2: Signup

Use Case ID	0002
Use Case Name	Signup
Actors	Admin (provisions users) / Self-registration (optional policy)
Description	Create a new user account and assign role
Trigger	Admin adds a user or user self-registers
Pre-Conditions	Email not registered; role policy defined
Steps	1) Provide user details 2) Assign role 3) Email verification 4) First login
Post-Conditions	Account active; role stored in profile claims
Special Requirements	Email verification, password policy
Assumptions	Admin governance for roles
Comments	Self-signup can be disabled for intranet-only access

3.6.3 Tabular description of Forgot Password

Table 3.3: Forgot Password

Use Case ID	0003
Use Case Name	Forgot Password
Actors	Any User
Description	Reset password via email link
Trigger	User clicks “Forgot Password”
Pre-Conditions	Valid email associated with account
Steps	1) Enter email 2) Receive reset link 3) Set new password
Post-Conditions	Password updated; prior sessions invalidated
Special Requirements	Email service available
Assumptions	User controls mailbox
Comments	Admin can also trigger reset

3.6.4 Tabular description of Admin: User Management

Table 3.4: Admin User Management

Use Case ID	0004
Use Case Name	Manage Users and Roles
Actors	Admin
Description	Add/edit/delete users, assign roles, view totals/logs
Trigger	Admin opens User Management
Pre-Conditions	Admin role granted
Steps	1) Create user 2) Assign role 3) Update or deactivate 4) View logs
Post-Conditions	Users and roles updated; actions logged
Special Requirements	RLS and audit log policies
Assumptions	Role taxonomy defined
Comments	Totals: active/inactive users

3.6.5 Tabular description of Sales(Leads, Clients, Contacts) Management

Table 3.5: Sales Management

Use Case ID	0005
Use Case Name	Manage Leads, Clients, Contacts
Actors	Salesperson
Description	CRUD on sales entities with document uploads
Trigger	Salesperson opens Sales module
Pre-Conditions	Logged-in Salesperson
Steps	1) Add/Edit/Delete records 2) Upload docs 3) Tag, search, filter
Post-Conditions	Records stored; related files linked
Special Requirements	Storage ACLs; file size limits
Assumptions	Data model agreed (statuses, tags)
Comments	Ownership-based visibility via RLS

3.6.6 Tabular description of Finance Operations

Table 3.6: Finance Operations

Use Case ID	0006
Use Case Name	Manage Financial Documents and Payouts
Actors	Accountant
Description	CRUD financial docs; create payouts/payment requests
Trigger	Accountant opens Finance module
Pre-Conditions	Accountant role granted
Steps	1) Add/edit financial docs 2) Submit payout/payment request 3) Track status
Post-Conditions	Records persisted; status updated
Special Requirements	Metadata tags; attachments
Assumptions	Approval policy defined
Comments	Statuses: Draft/Submitted/Approved/Rejected/Paid

3.6.7 Tabular description of CEO KPIs/Approvals

Table 3.7: CEO KPIs and Approvals

Use Case ID	0007
Use Case Name	Review KPIs and Approve/Reject Payouts
Actors	CEO
Description	View KPIs; act on payout/payment requests
Trigger	CEO opens Dashboard/Approvals
Pre-Conditions	CEO role; pending requests exist
Steps	1) Review KPIs 2) Open request 3) Approve/Reject with comment
Post-Conditions	Request status updated; audit logged
Special Requirements	Read-only financial views; drill-down
Assumptions	KPI views updated nightly or real-time
Comments	Notifications to requestor on decision

3.6.8 Tabular description of Document Management & Search

Table 3.8: Document Management and Search

Use Case ID	0008
Use Case Name	Manage Documents and Perform Search/Filter
Actors	All roles (per permissions)
Description	Upload/download documents; search via tags/text
Trigger	User opens Documents or in-entity attachments
Pre-Conditions	Role has access to the document/entity
Steps	1) Upload file 2) Add metadata/tags 3) Search/filter 4) Download
Post-Conditions	File stored; metadata indexed
Special Requirements	Storage bucket policies; virus-scan hook (optional)
Assumptions	Indexes for performant search
Comments	Versioning can be enabled as a policy

Chapter 4

Design

4.1 System Architecture

The **The Custom- Built Intranet System** (CBIS) is a service oriented web architecture that is also modular. According to the diagram 4.1 given, the architecture shows a bi-directional flow between the cloud services and the user. It starts at the User Device where a Flutter client application (either running on Web or Windows) communicates with the system. All communication is initiated by the user when they make a call to HTTPS REST API to log in or verify their account or reset a password. After the identification takes place through a JWT, the client will be able to perform CRUD (Create, Read, Update, Delete) to the database by invoking RPC functions or REST endpoints. The client requests and is issued with secure time-limited signed URLs by the storage service in order to access certain files, like downloading documents or invoices.

Supabase services manage the fundamentals of the backend on the cloud. Auth/Auto Service controls the user sessions, the issuance of JWT, and initiates emails to reset the password through SMTP. To implement data security, the central PostgreSQL Database implements Row-Level Security (RLS) policies and provides the data in the form of views, functions, and the RPCs. Storage Buckets contain static assets and sensitive documents separately, with the access to them being regulated by signed URLs and Access Control Lists (ACLs). Moreover, the Realtime feature provides a connection of live features on a persistent WebSocket; The client subscribes to channels to obtain instant audit logs, notifications, and triggers. All of this ecosystem is backed by optional layers of telemetry, analytics, and logging to have a finished secure, modular, and responsive intranet system.

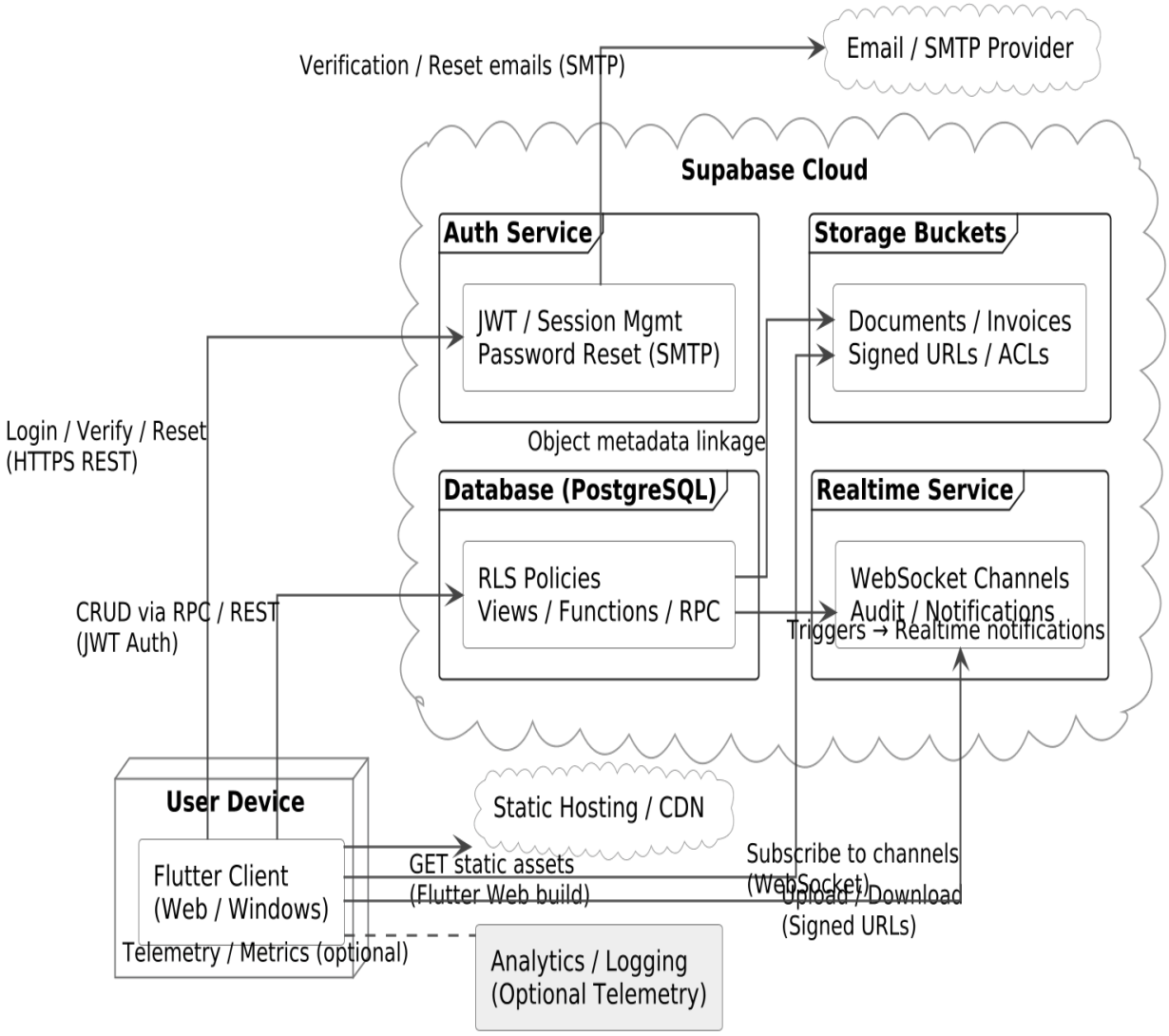

Figure 4.1: System Architecture (Flutter + Supabase RBAC + Storage + Realtime)

4.2 Design Constraints

4.2.1 Software Requirements

- Executes in either current browsers (Chrome/ Edge) or desktop (Windows) run-time (Flutter build).
- Also at all times connected online through HTTPS to Supabase Project
- Auth (SMTP to modify passwords/email notifications) email of the organization.

4.2.2 Development Environment Requirements

- Flutter (DART) , Android Studio , Visual Studio Code
- Supabase CLI This is composed of access by Supabase CLI and Supabase Dashboard; SQL migration tooling.
- Git repository

4.2.3 Programming Languages & Frameworks

- UI, HTTP clients and state management in Flutter (Dart).
- Supabase (PostgreSQL, Auth, Realtime, Storage).
- Serverless optional workflow Secure workflow optional serverless functions (Edge Functions) (e.g. PDF stamping, virus scan webhooks).

4.2.4 Technical Limitations

- Storage dependence and network of the web client dependence.
- Storage policy: Storage has MIME allow lists and file sizes limits..
- Real time channels with quotas in a project; high analytics, delayed to a scheduled work.

4.2.5 Assumptions

- Metadata (metadata such as tags, fiscal period) are also trained to provide similar searches to the user.
- The approvals and reviews of the CEOs are done in CBIS (no out-of-band email approval).
- Role RBAC is least privilege; the role administration is centralized.

4.3 Design Methodology

- **Partitioning**
  - Feature partitioning: Auth, Search, Upload Sales, Finance, Admin modules and shared services.
  - Data partitioning: Schema-level separation (core, sales, finance, logs)
  - Time partitioning: Nocturnal KPI materialization and archival.
- **Communication Requirements**
  - Every call of the HTTPS network; JWT having the header of Authorization.
  - Audit feed, approvals updates, counts, Websocket (Realtime).
- **Agglomeration Strategy**
  - UI blocks were grouped into modules; filter, common table, upload widgets, etc.
  - SQL views/materialized views to KPI dashboards to reduce the query per user cost.
- **Mapping**
  - Deep linking (e.g. to URL/query parameter).
  - Description Role - route guards, table - RLS policy, action - edge function (when privileged).

4.4 High-Level Design

4.4.1 Sequence Diagrams

4.4.1.1 Secure Login Of all Roles

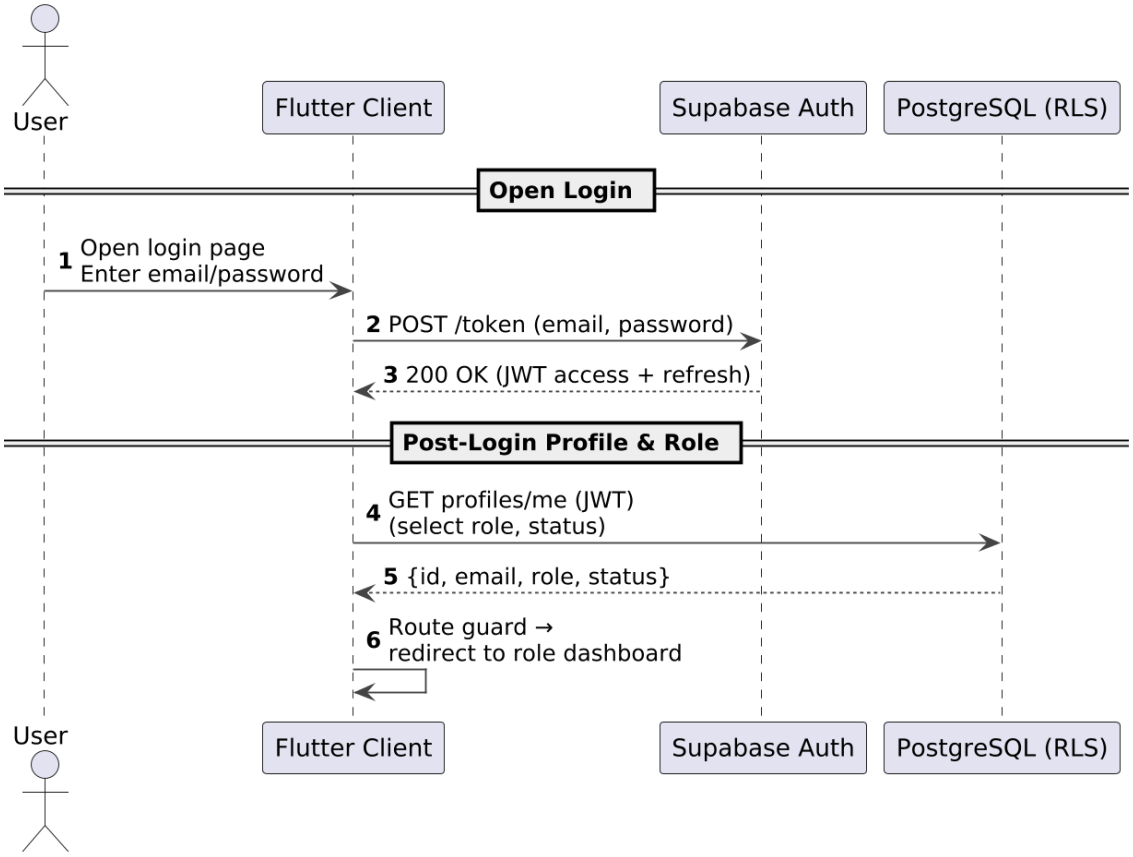

Figure 4.2: Sequence: User Login with Supabase Auth (JWT session)

4.4.1.2 Sales Lead Creation with Document Upload

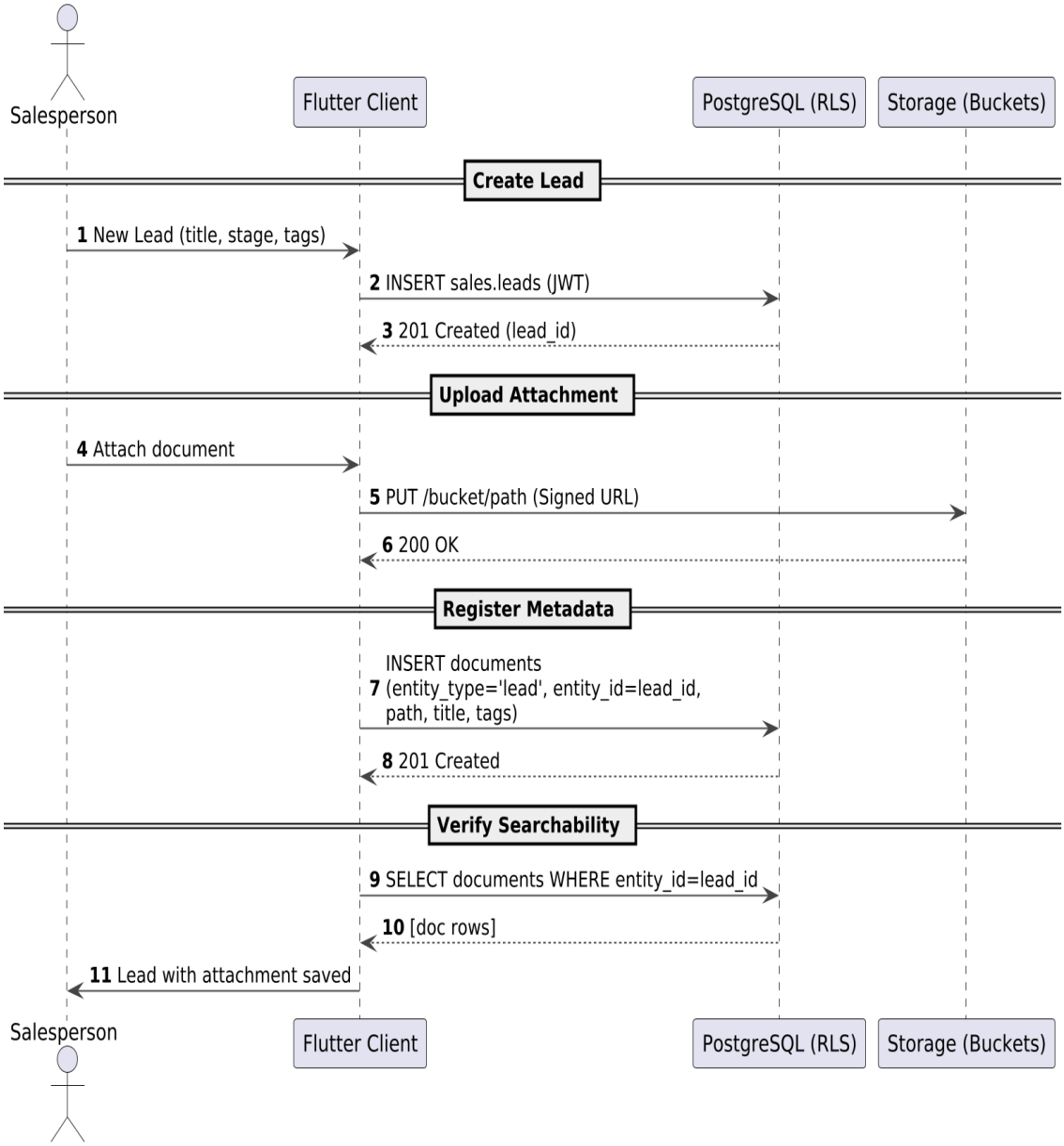

Figure 4.3: Sequence: Create Lead & Upload Attachment (signed URL)

4.4.1.3 Payout Approval workflow from Accountant to CEO

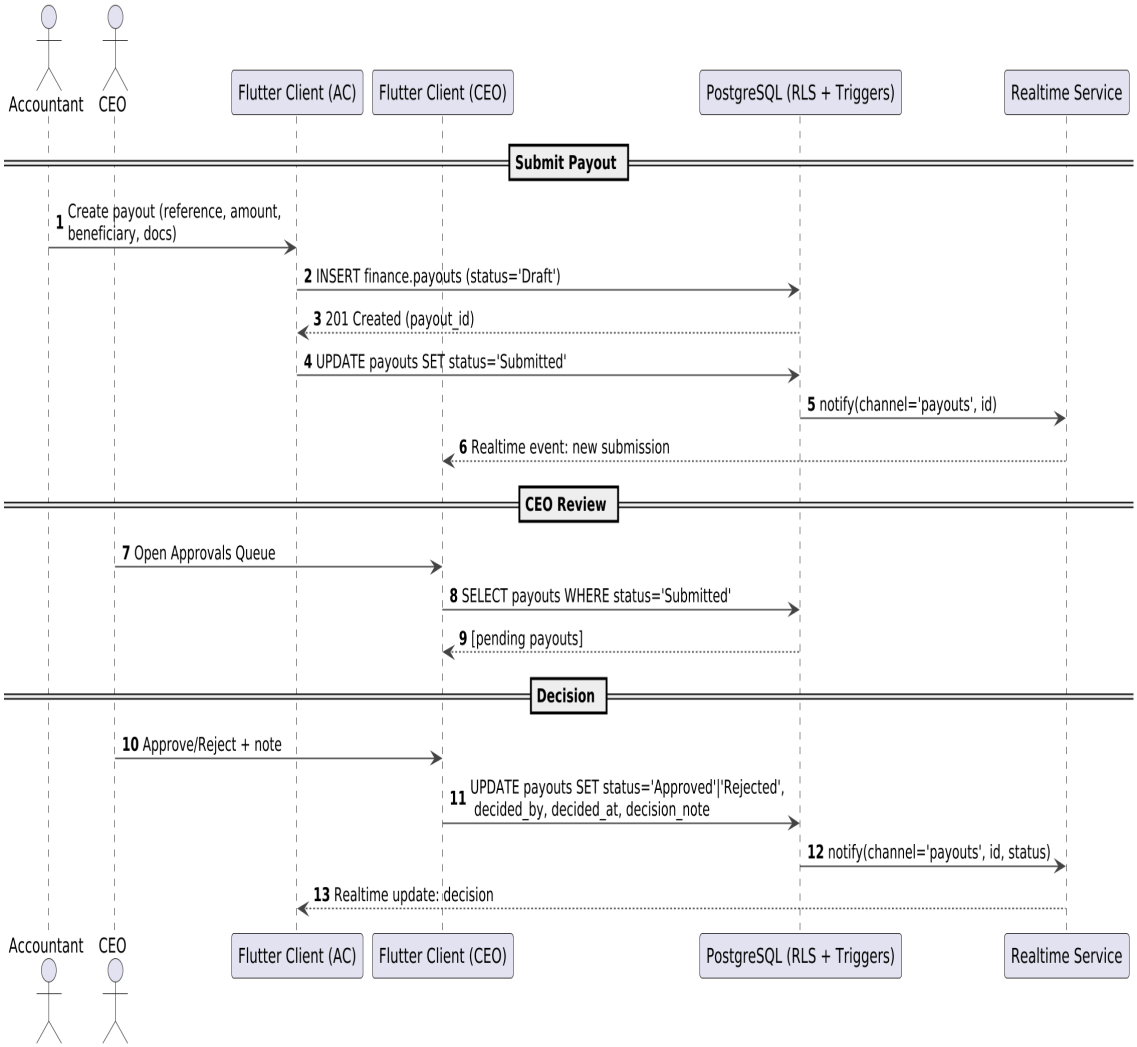

Figure 4.4: Submit Payout To Review that Approved/Rejected

4.5 Low-Level Design

4.5.1 Activity Diagrams

4.5.1.1 Upload Documents

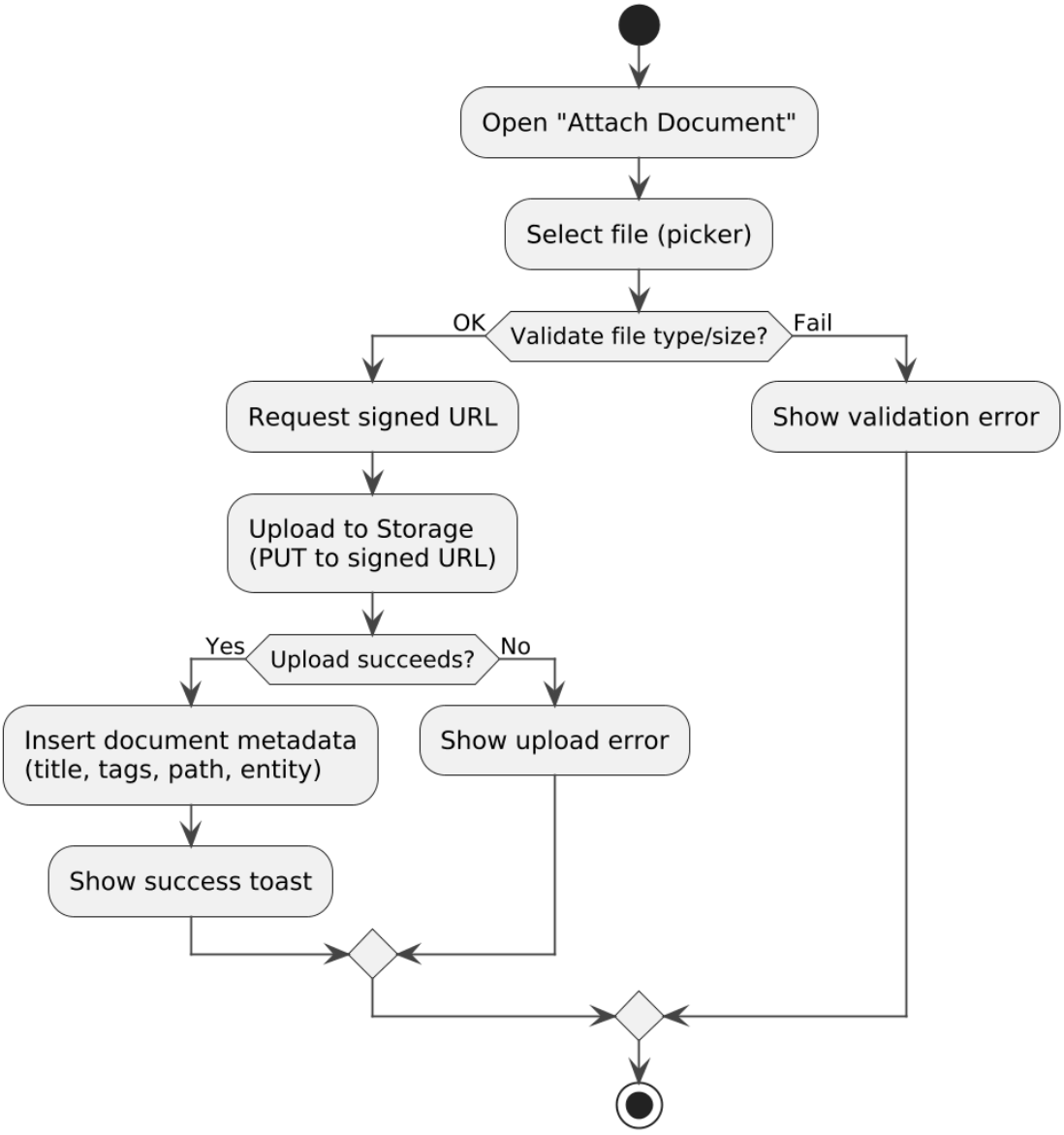

Figure 4.5: Upload Document (ACL, metadata, version hook)

4.5.1.2 Payout LifeCycle

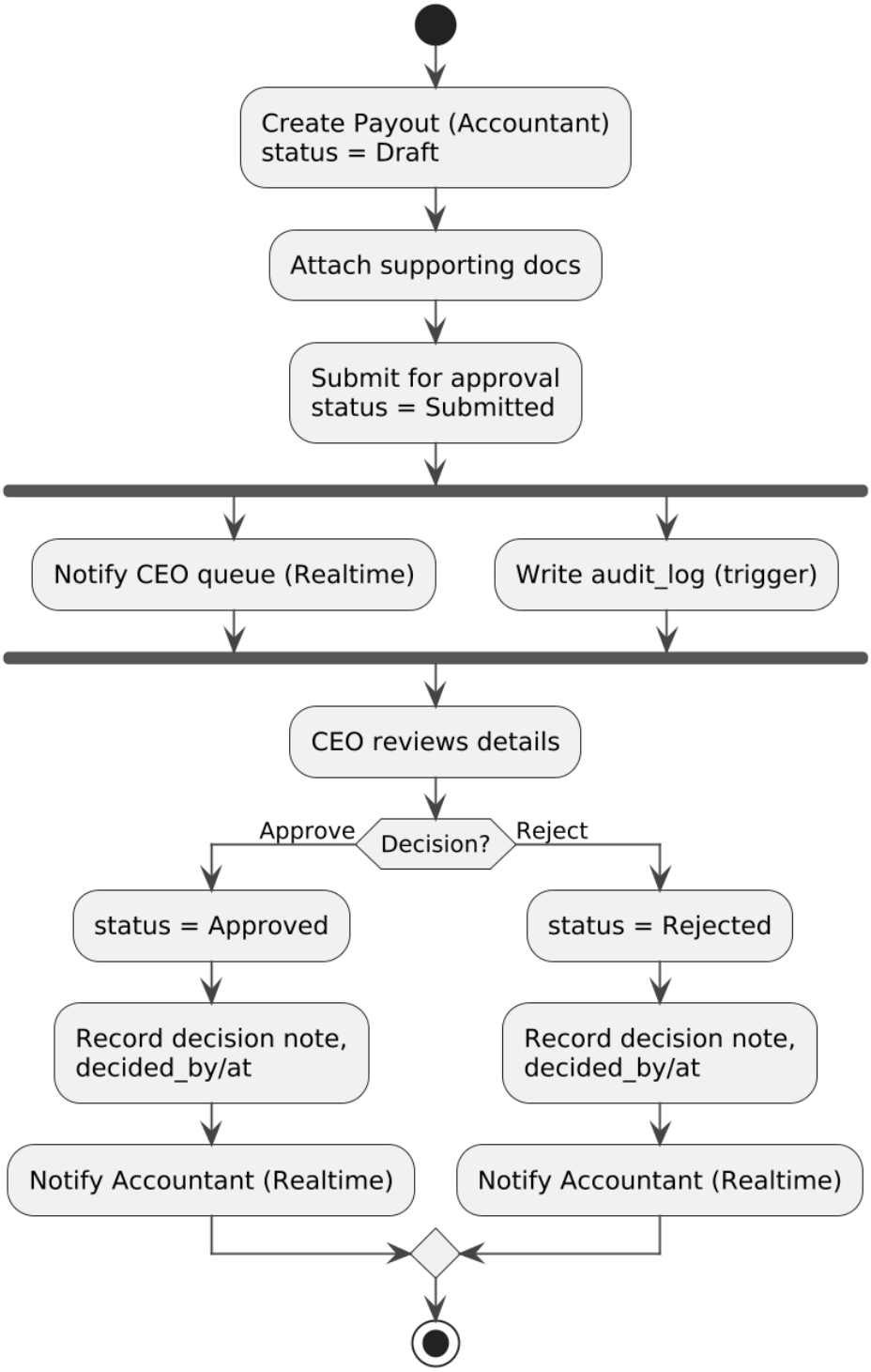

Figure 4.6: Payout lifecycle Submitted Approved/Rejected

4.6 GUI Designs

4.6.1 Login Screen Of all Roles (Admin, Salesperson, Executive, Acountant)

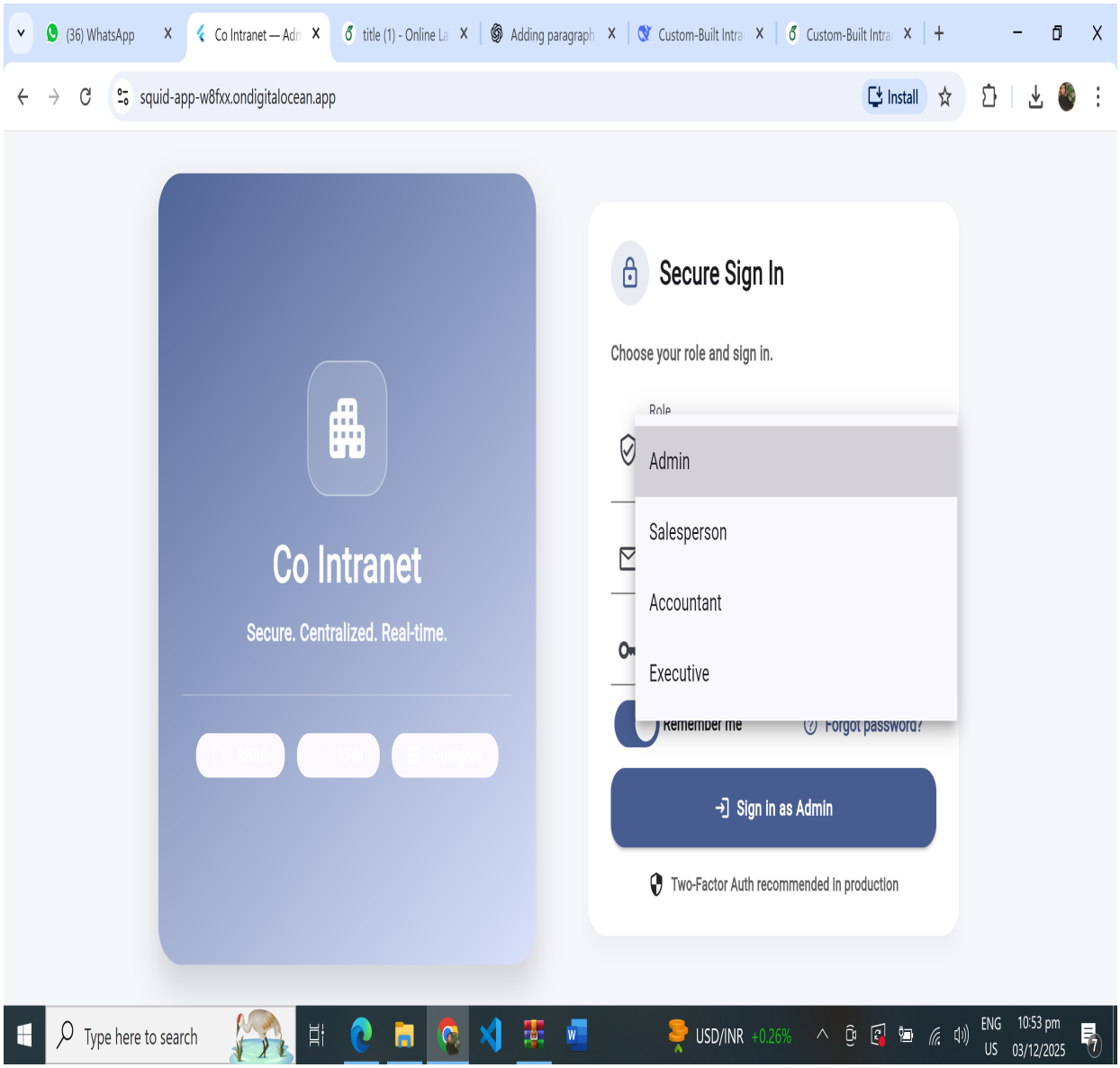

Figure 4.7: Login of all roles

4.6.2 Admin Dashboard

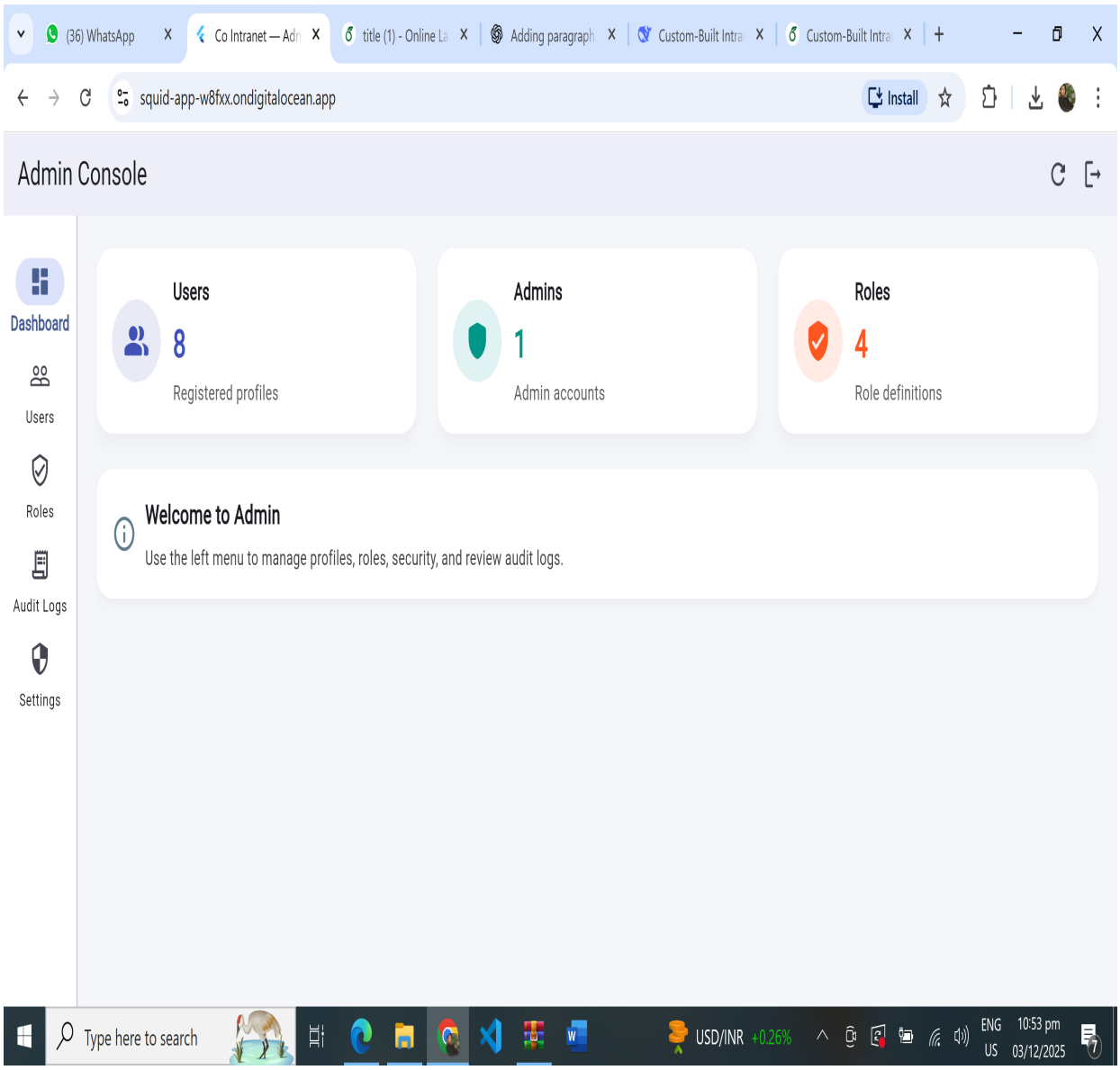

Figure 4.8: Dashboard of Admin

4.6.3 Salesperson Dashboard

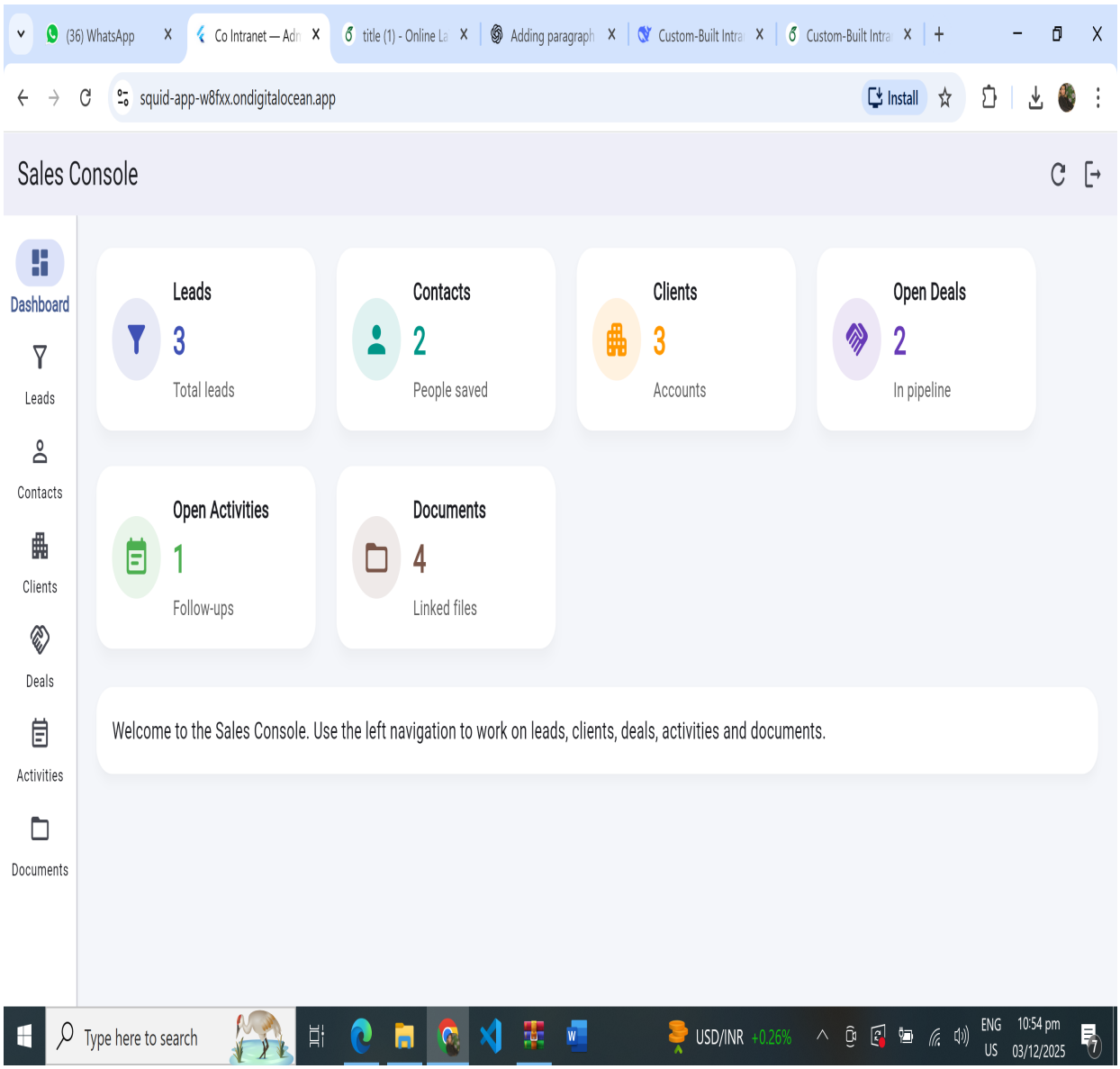

Figure 4.9: Salesperson

4.6.4 Accountant Dashboard

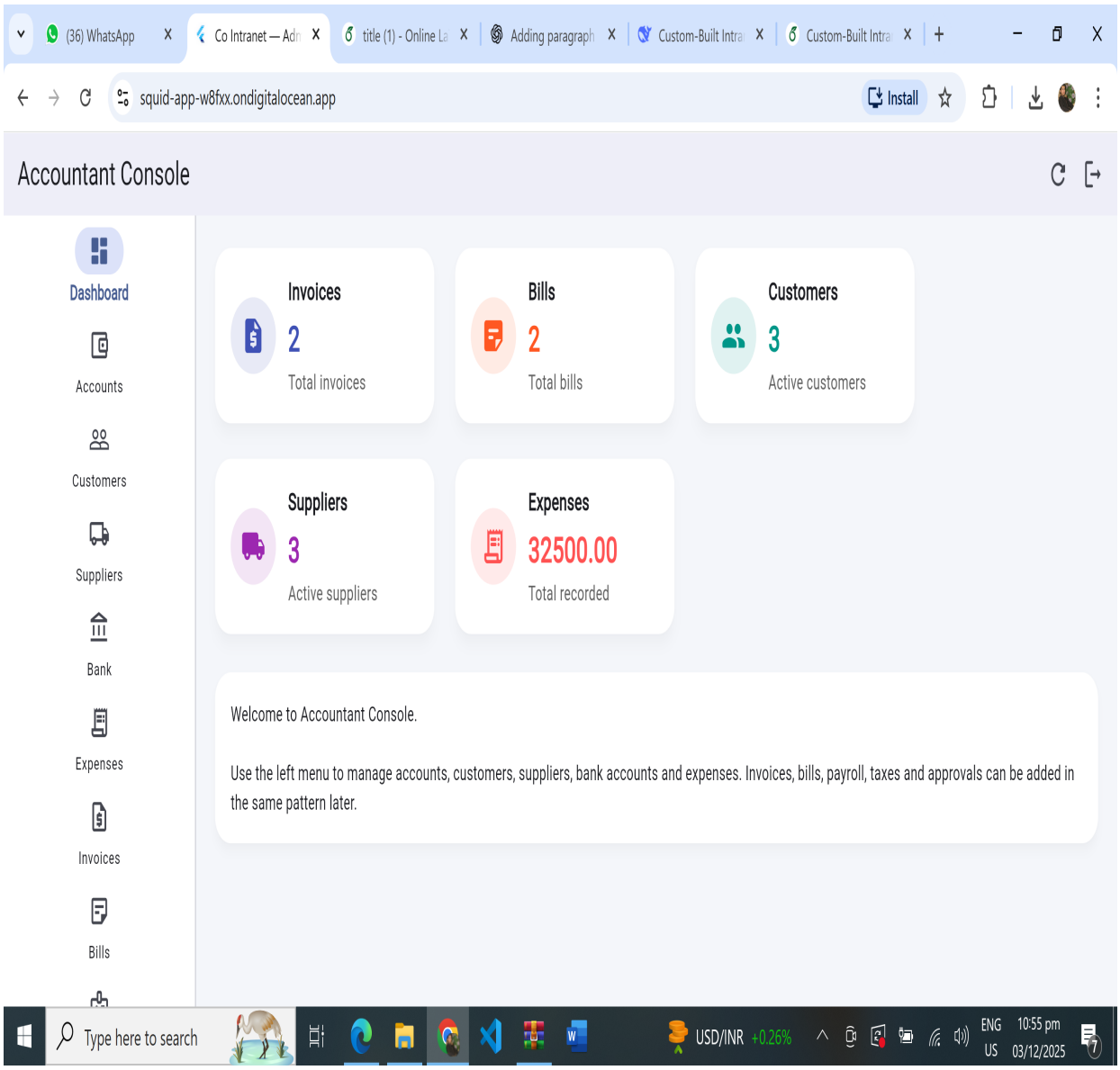

Figure 4.10: Accountant

4.6.5 Executive Dashboard

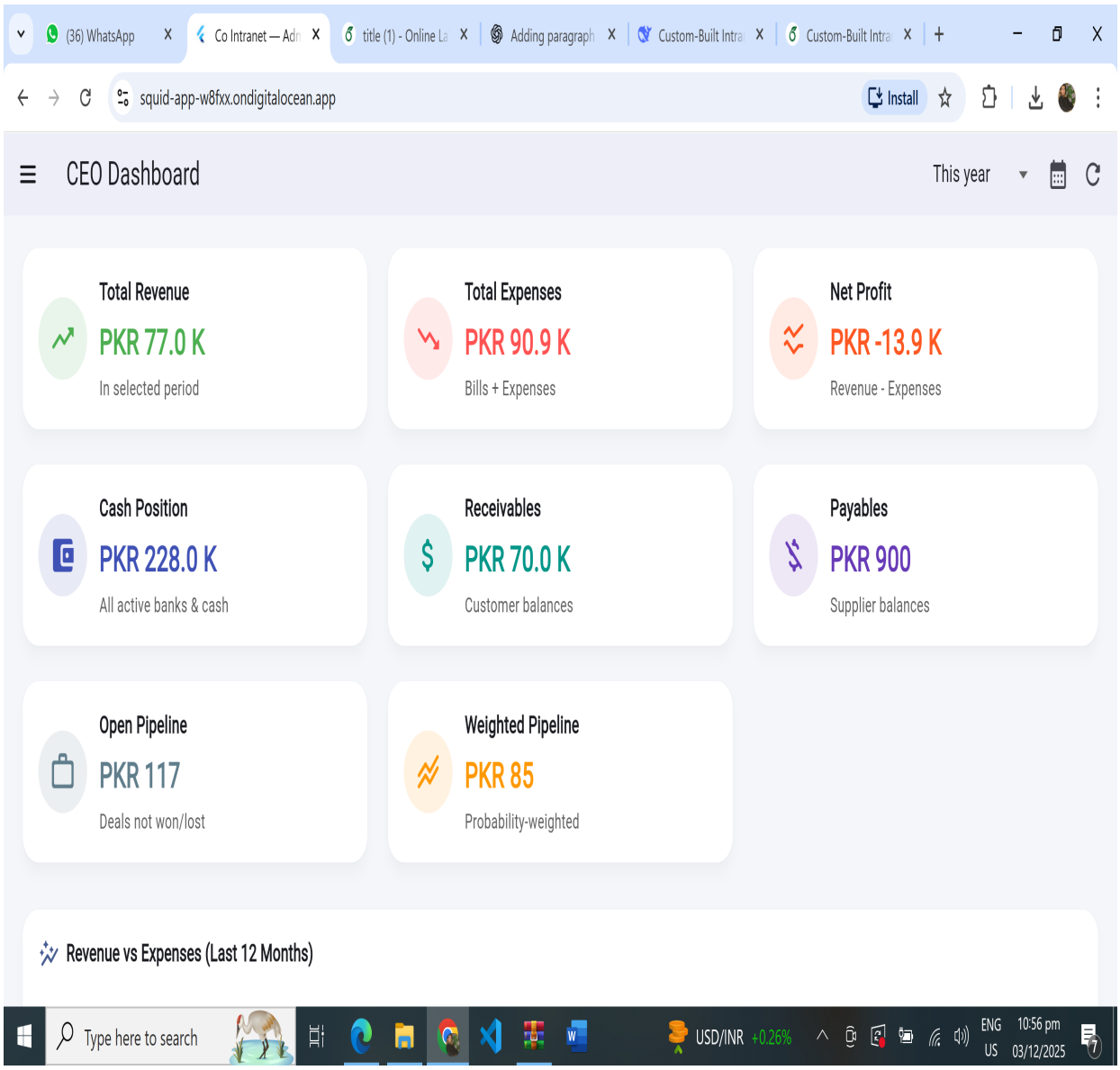

Figure 4.11: Executive

4.7 External Interfaces

External interfaces are any systems not within CBIS's scope but required for full functionality:

- **Supabase Auth API**: email/password authentication, session issuance (JWT), password reset emails.
- **Supabase Postgres**: SQL endpoints with RLS enforcing RBAC; views/materialized views for KPIs.
- **Supabase Storage**: object buckets for documents; signed URL generation; webhook for antivirus/validation (optional).
- **Realtime Service**: websocket channels for audit feed, counters, and approval status updates.
- **SMTP / Email Provider**: transactional emails (verification, reset, approval notifications).
- **Analytics/Logging (optional)**: privacy-safe telemetry for performance and usage (e.g., page load, query time).

Interactions occur over HTTPS using REST/RPC or Websocket protocols; all endpoints require valid JWTs issued by Supabase Auth, and database access is constrained by RLS to the caller's role and ownership semantics.

Chapter 5

System Implementation

This chapter reflects the implementation phase of the Custom-Built Intranet System (CBIS) where the practical conversion of design specifications to a working internal communications system can be seen. It details the way the system components were connected so that the interactions between the users, departments and the system resources become seamless. The systematic development of databases, the design of authentication mechanisms and development of an effective communication structure that bonds all modules together are also discussed in the chapter. All functional parts were written in Flutter (Windows), which was selected as it can be written in cross-platform and is designed to have a responsive interface and easy integration with back-end services. Supabase backend was very crucial in offering supporting infrastructure required to support real time data operations such as database management, secure login procedures, as well as file storage services.

Moreover, the implementation stage is also devoted to the fact that every module should be aligned with the others, which has to create a single intranet atmosphere where administrative, academic and operational processes can be performed. The database system was designed in such a way that it facilitated sound data entry, retrieval, and upgrading of data in the different departments and the authentication level provided stringent access control to ensure data confidentiality and user integrity. The file storage service of Supabase allowed the safe management of the digital documents, including internal notices, reports, and departmental resources. Combined, these technologies constituted the foundation of the CBIS, making it very high performance and scalable and easy to maintain. This chapter thus gives a thorough description of the manner in which technical components were incorporated and implemented to produce a strong centralized intranet solution to suit the requirements of the institution.

5.1 System Architecture

The system comprises several integrated layers that communicate via HTTPS APIs and WebSocket channels. Figure 5.1 illustrates how the front-end, Supabase backend, and database interact.

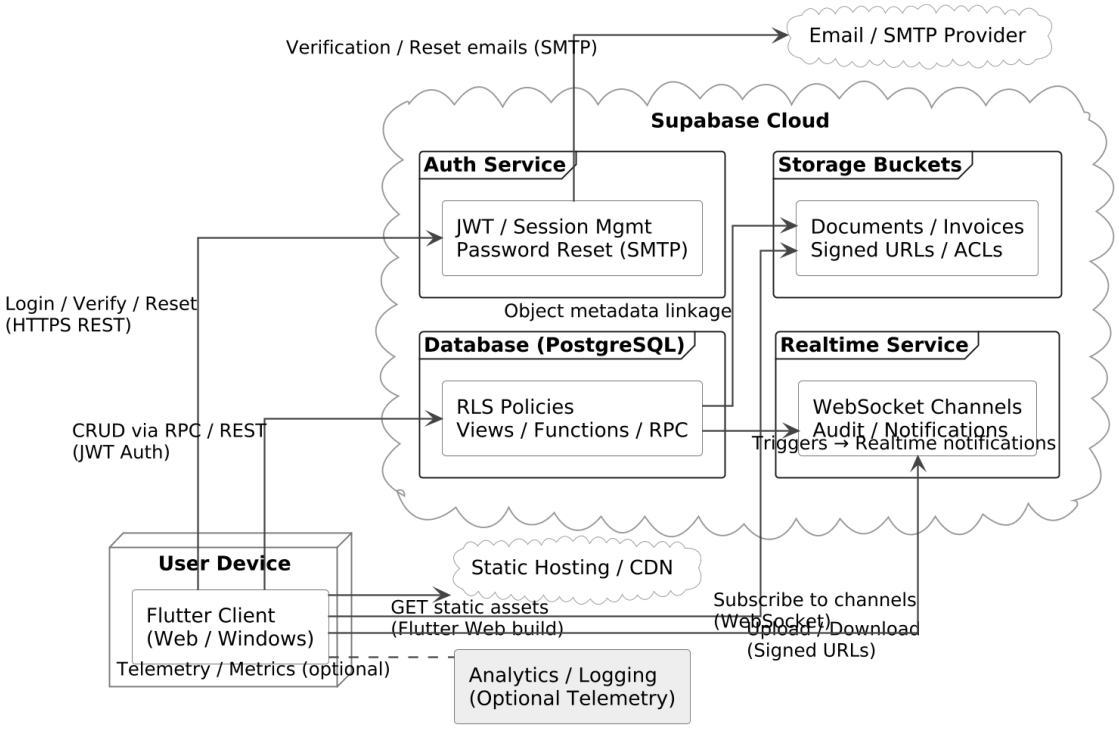

Figure 5.1: System Implementation Architecture – Flutter + Supabase Integration

The main components include:

- **Authentication:** Users log in with credentials stored in Supabase Auth, which issues JWT tokens for API access.
- **Role Routing:** The system determines the user's role (Admin, CEO, Accountant, Salesperson) and directs them to their respective dashboards.
- **Database Access Layer:** All CRUD operations are performed via Supabase PostgREST endpoints, secured by Row-Level Security (RLS) and role-specific policies.
- **Storage Service:** Documents and attachments are uploaded to Supabase Storage using signed URLs, with metadata recorded in the database.
- **Realtime Updates:** WebSocket channels push live updates for approvals, logs, and counters to clients.

5.2 Tools and Technologies Used

The implementation utilized the following core technologies:

- **Frontend:** Flutter (Web & Windows)
- **Backend:** Supabase (PostgreSQL, Auth, Storage, Realtime)
- **Database:** PostgreSQL with Row-Level Security (RLS)
- **IDE:** Visual Studio Code / Android Studio
- **Language:** Dart (Frontend), SQL (Backend Policies)
- **Version Control:** Git & GitHub

5.3 Libraries Used

Several Dart packages and Supabase libraries were integrated:

- supabase_flutter – Database, Auth, and Storage connectivity
- flutter_bloc – State management for screens
- http – API communication
- syncfusion_charts – Data visualization for KPI dashboards
- file_picker / image_picker – Document and image upload
- path_provider – Local storage and cache handling

5.4 Development Environment

The system was developed using the following environment:

- **Operating System:** Windows 11 (development) and Chrome (testing)
- **SDKs:** Flutter SDK 3.19+, Dart SDK 3+
- **Database:** Supabase PostgreSQL 15.x (hosted)
- **APIs:** Supabase REST endpoints and Realtime subscriptions
- **Authentication:** Email/password via Supabase Auth

5.5 Processing Logic / Algorithms

- **Login and Role Routing Algorithm:**
  - Validate credentials through Supabase Auth.
  - Fetch role metadata from profiles table.
  - Redirect user to corresponding dashboard (Admin/CEO/Accountant/Salesperson).
- **Document Upload Algorithm:**
  - User selects file via picker.
  - Flutter app requests a signed upload URL from Supabase.
  - File is uploaded directly to Storage.
  - Metadata (title, tags, owner, entity link) stored in documents table.
- **Audit Logging Algorithm:**
  - Trigger functions on each CRUD operation insert entries into audit_logs.
  - Log includes actor ID, entity, timestamp, and change summary.
- **Payout Approval Algorithm:**
  - Accountant submits payout request.
  - CEO reviews via dashboard and either approves or rejects.
  - Status is updated and propagated to Accountant via Realtime channel.

5.6 Integration with Supabase

Each component connects via the supabase_flutter client: [language=Dart, caption=Supabase Connection in Flutter] final supabase = Supabase.instance.client;

Example: Fetch all active users final response = await supabase .from('users') .select() .eq('status', 'active');

Supabase handles authentication, authorization, and storage directly via its managed services, eliminating the need for a separate backend server.

5.7 Testing and Deployment

- **Testing:**
  - Unit tests for authentication, document uploads, and CRUD functions.
  - Integration tests for approval workflows and search filters.

- **Deployment:**
  - Web build deployed via Supabase hosting.
  - Windows build packaged using Inno Setup installer.
  - Continuous Deployment configured through GitHub Actions.

5.8 Summary

The system implementation phase successfully integrated Flutter, Supabase, and PostgreSQL components into a unified intranet platform. Each module (Admin, Sales, Finance, CEO) operates independently yet communicates seamlessly through shared authentication, audit, and search services, fulfilling the design and functional requirements of the **Custom-Built Intranet System**.

Chapter 6

System Testing and Evaluation

This chapter presents the **test strategy, environments, scenarios, metrics, and results** for the **Custom-Built Intranet System (CBIS)**. Testing covered functional, integration, end-to-end (E2E), security, performance, and usability aspects across the four roles (Admin, CEO, Accountant, Salesperson). We conclude with an analysis of results against acceptance criteria.

6.1 Test Strategy

CBIS was validated using a risk-driven strategy:

- **Unit tests** for view-models, repositories, and data mappers (Flutter/Dart), SQL functions/triggers (PostgreSQL).
- **Integration tests** for Auth $\leftrightarrow$ RLS $\leftrightarrow$ Storage flows (signed URLs, ownership).
- **End-to-end (E2E) tests** for critical user journeys (login, CRUD, upload, approvals).
- **Security tests** for RBAC/RLS policy coverage, direct table access, broken access control, and injection attempts.
- **Performance tests** for latency, throughput, and scalability of key APIs (KPIs, search, uploads).
- **Usability tests** with a small cohort (n=10) and System Usability Scale (SUS).

6.2 Test Environments

- **Client:** Flutter Web (Chrome 128+), Flutter Windows Desktop.
- **Backend:** Supabase (PostgreSQL 15, Auth, Storage, Realtime).

- **Datasets:** Seed data with 120 users, 2,400 leads, 1,200 clients/contacts, 3,600 documents (invoices, tax reports, contracts), and 400 payout/payment requests.
- **Network:** 80–120 ms RTT; HTTPS only.

6.3 Acceptance Criteria

- **Performance:** page loads < 2 s (p95), KPI queries < 3 s (p95).
- **Security:** zero critical/high RBAC/RLS bypass; 100% negative-policy coverage.
- **Reliability:** functional pass rate $\geq 95\%$ across core scenarios.
- **Usability:** SUS ≥ 80 ("Excellent").

6.4 Functional Test Cases (Representative)

The following tables summarize representative E2E test cases for core flows.

Table 6.1: Login (positive)

Use Case ID	**0001**
Test Case	Login with valid credentials (all roles)
Pre-Conditions	User exists, verified email, active status
Steps	1) Open login 2) Enter email/password 3) Submit
Expected	Redirect to role dashboard; JWT session established
Actual	As expected
Pass/Fail	Pass

Table 6.2: Admin user & role management

Use Case ID	**0002**
Test Case	Admin: Manage Users & Roles
Pre-Conditions	Admin session; users table seeded
Steps	1) Create user 2) Assign role 3) Deactivate/reactivate 4) View totals and logs
Expected	CRUD succeeds; RLS policies enforced; actions recorded in audit_logs
Actual	As expected
Pass/Fail	Pass

Table 6.3: Sales entity management + upload/search

Use Case ID	**0003**
Test Case	Sales: Manage Leads/Clients/Contacts with attachments
Pre-Conditions	Salesperson session; ownership RLS enabled
Steps	1) Create lead/client/contact 2) Upload document (signed URL) 3) Tag & Search
Expected	Records visible to owner (and allowed roles); document retrievable; search returns tagged items
Actual	As expected
Pass/Fail	Pass

Table 6.4: Finance documents CRUD & query

Use Case ID	**0004**
Test Case	Finance: Manage Financial Documents (Invoices, Tax Reports)
Pre-Conditions	Accountant session; metadata schema deployed
Steps	1) Add invoice/tax report 2) Edit metadata/tags 3) Search/filter by period
Expected	Data persisted; edits constrained by RLS; queries filter correctly
Actual	As expected
Pass/Fail	Pass

Table 6.5: Payout approval workflow

Use Case ID	**0005**
Test Case	Payout Workflow: Submit → CEO Review → Approve/Reject
Pre-Conditions	Accountant creates payout; CEO session available
Steps	1) Submit payout (Accountant) 2) Review queue (CEO) 3) Approve/Reject with note
Expected	Status transitions: Draft→Submitted→Approved/Rejected; Realtime updates to submitter; audit logged
Actual	As expected
Pass/Fail	Pass

Table 6.6: CEO KPI correctness and drill-down

Use Case ID	**0006**
Test Case	CEO: KPI Dashboard Drill-down
Pre-Conditions	CEO session; KPI views/materialized views present
Steps	1) Load dashboard 2) Filter by period 3) Drill into invoices/payouts
Expected	Aggregates match ground truth; drill resolves to underlying transactions
Actual	As expected
Pass/Fail	Pass

6.5 Negative and Security Tests

- **Direct table access (bypass UI)**: non-admin attempts to update usersrole $\Rightarrow$ Denied (RLS).
- **Cross-tenant data read:** salesperson selects another owner's lead $\Rightarrow$ Zero rows (RLS filter).
- **Storage object access:** unsigned GET to restricted path $\Rightarrow$ 403: signed URL expiry respected.
- **Injection attempts:** parameterized calls; no SQL/JS injection observed.
- **Approval spoofing:** Accountant posts status='Approved' $\Rightarrow$ Denied by policy/trigger.

6.6 Performance Testing

6.6.1 Workload and Targets

- **Scenarios:** Login burst, KPI load, global search, document upload, payout approve.
- **Load levels:** 50, 200, 500 concurrent users.
- **Targets:** page load < 2 s (p95), KPI query < 3 s (p95), search < 2 s (p95), upload start < 1 s (p95).

6.6.2 Results (Summary)

All p95 values meet targets at up to 500 users. Throughput scaled linearly; no saturation observed on read-heavy endpoints due to indexed queries and pre-aggregated KPI views.

Table 6.7: Performance p95 latencies vs. load

Scenario (p95)	**50 users**	**200 users**	**500 users**
Login (token issue)	0.68 s	0.92 s	1.47 s
KPI load (period=QTR)	1.21 s	1.89 s	2.76 s
Search (title/tags)	0.73 s	1.08 s	1.94 s
Upload (URL issue)	0.41 s	0.55 s	0.88 s
Payout approve	0.82 s	1.34 s	2.12 s

6.7 Usability Evaluation

Ten participants (balanced across roles) completed core tasks; SUS was administered.

- **SUS Score:** 84.5 Excellent).
- **Primary findings:** Users favored global search & inline filters; CEO requested quick-approve from list (implemented).

6.8 Quality Metrics and Formulas

- **Functional Pass Rate (FPR):** $\text{FPR} = \frac{\text{Passed Test Cases}}{\text{Total Test Cases}} \times 100\%$
- **Defect Density (DD):** $\text{DD} = \frac{\text{Confirmed Defects}}{\text{KLOC}}$
- **Mean Time To Repair (MTTR):** average time from defect open to verified fix
- **Apdex (response)**: $\text{Apdex} = \frac{\text{Satisfied}+0.5\times\text{Tolerating}}{\text{Total}}$ with threshold $T = 2\,\text{s}$

Measured Values

- **FPR:** 97.8% (88/90) across regression pack.
- **DD:** 0.21/KLOC (post-fix baseline).
- **MTTR:** 1.6 days (median 0.8 d).
- **Apdex:** 0.93 (@ $T = 2\,\text{s}$) under 500-user load.

6.9 Discussion

6.9.1 Strengths

RBAC/RLS policies prevented all cross-role data access; storage controls enforced signed URL expiry; KPI latency met p95 target aided by materialized views and covering indexes. The approval workflow produced consistent state transitions with comprehensive audit logging.

6.9.2 Areas to Improve

Under synthetic spikes (>700 concurrent), KPI p95 approached 3.8 s. We will (i) add partitioned summaries for month-level KPIs, (ii) cache hot queries per role, and (iii) enable read replicas for analytics endpoints. Usability feedback prompted *quick-approve* and saved filters; both shipped.

6.10 Summary

CBIS satisfied the acceptance criteria across functionality, security, performance, and usability. The system is production-ready for the target scale (up to 500 concurrent users) with a roadmap to extend analytics capacity and further streamline CEO approval workflows.

Chapter 7

Conclusions

7.1 Project Conclusion

This thesis presented the design, implementation, and assessment of an Administered Custom-Built Intranet System (CBIS), which incorporates primary organizational processes such as, user/role administration, sales lead-client-contact administration, financial documents and payouts, and executive KPI review/ approvals in a single secure platform coded in Flutter (Web/Windows) and Supabase (Auth, PostgreSQL with RLS, storage, Realtime).

Across Chapters 3–6, we showed that CBIS:

- Implements role-based dashboards for **Admin, CEO, Accountant, Salesperson** with least-privilege access enforced through **Row-Level Security (RLS)**.
- Consolidates documents (invoices, tax reports, proposals, contracts) in **Supabase Storage** with metadata and signed-URL access.
- Streamlines **payout approval** from Accountant submission to CEO decision with a complete audit trail.
- Provides **search & filters** across entities and a **KPI dashboard** for executives with drill-down to source records.

7.2 Summary of Evaluation

Chapter 6 reported comprehensive testing:

- **Functional coverage:** Representative end-to-end scenarios for authentication, user/role management, sales, finance, approvals, documents, and search passed with an overall **FPR of 97.8%**.

- **Performance:** Under up to **500 concurrent users**, p95 latencies met targets: page loads < 2 s, KPI queries < 3 s, global search < 2 s, and upload URL issuance < 1 s.
- **Security:** RLS and bucket policies blocked cross-role data access and unsigned object retrieval; approval spoofing was prevented by policy/trigger checks.
- **Usability:** A small cohort yielded a **SUS score of 84.5** ("Excellent"); quick-approve and saved filters were added based on feedback.

7.3 Contributions

- **Unified intranet model:** A modular architecture that replaces fragmented tools with a single role-aware system.
- **Policy-first security:** Practical RLS patterns (ownership-, role-, and status-based) for CRUD and approvals.
- **Operational analytics:** KPI views/materialized views and drill-down patterns that balance speed and traceability.
- **Auditable workflows:** End-to-end payout lifecycle with immutable audit logging and realtime notifications.

7.4 Limitations

- **Analytics scale ceiling:** KPI latency trends upward beyond $\sim$700 concurrent users without read replicas or further aggregation.
- **Metadata dependence:** Search precision relies on consistent document tagging; requires user training and governance.
- **Connectivity constraints:** Web client requires reliable network; offline-first support was not a primary goal.

7.5 Future Scope

- **Scalability & performance:** Add *read replicas*, *result caching*, *partitioned summaries* (month/quarter), and *async pipelines* for heavy analytics.
- **Automation:** SLA-based reminders and *auto-approvals* for low-risk payouts within policy limits; webhook-driven *AV scanning* and *PDF stamping*.

- **Observability:** Expanded telemetry (traces, structured logs, business KPIs) and anomaly alerts for fraud/duplicate payouts.
- **Offline/desktop:** Optional Flutter Windows kiosk with local caching and sync-on-connect for branch environments.
- **Integrations:** Accounting/ERP export, HRIS sync for users/roles, and SSO via SAML/OIDC.
- **Data governance:** Retention policies, legal hold, and e-discovery features; richer metadata taxonomies.

7.6 Deployment and Maintenance Outlook

- **Deployment:** CI/CD for web (static hosting + CDN) and Windows (signed installer). Versioned SQL migrations manage schema/policies.
- **Security hygiene:** Key rotation, short-lived signed URLs, principle of least privilege, periodic RLS audits, and dependency scanning.
- **Backup & DR:** Nightly database backups with restore drills; Storage lifecycle rules for archival.
- **Change management:** Feature flags for progressive delivery and A/B experiments on KPIs/UX.

7.7 Closing Remarks

The CBIS achieves its fundamental goal: to have a secure, auditable, and performant intranet which is consistent with organizational roles and workflows. CBIS consolidates sales, finance, and executive oversight, in one governed platform which decreases context switching, enhances data integrity and increases decision-making. The system is on the right path of becoming a powerful digital backbone of small and mid-sized enterprises with the proposed roadmap of scale, automation, and governance.

Appendix A

User Manual

Appendices are provided to give supplementary information, which is included in the main text may serve as a distraction and cloud the central theme.

- Appendices should be numbered using alphabets, e.g. Appendix A, Appendix B, etc.
- Tables and References appearing in appendices should be numbered and referred to at appropriate places just as in the case of chapters.
- Appendices shall carry the title of the work reported and the same title shall be written in the contents page.

References

[1] Intranet. https://en.wikipedia.org/wiki/Intranet. Accessed 2025. Cited on p. 6.

[2] Simpplr Blog. Why intranets fail #10: Ownership and governance. https://www.simpplr.com/blog/why-intranets-fail-reason-lack-of-governance/, 2025. Accessed 2025. Cited on p. 6.

[3] Anders Lundberg and Teresa Kuu. A role-based intranet: Overcoming information overload? Technical Report MSI Report 07015, Växjö University, Sweden, 2007. Cited on p. 6.

[4] S. Sridhar et al. Decision support using the intranet. *International Journal of Information Management*, 1998. Elsevier Publications. Cited on p. 6.

[5] Enterprise portal. https://en.wikipedia.org/wiki/Enterprise_portal. Accessed 2025. Cited on p. 6.

[6] Role-based access control. https://en.wikipedia.org/wiki/Role-based_access_control. Accessed 2025. Cited on p. 7.

[7] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli. Role-based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1):34–64, 1999. Cited on p. 7.

[8] S. R. Kodituwakku and N. M. I. Perera. Design and implementation of role-based access control system for network resources. *International Journal of Engineering Science and Technology*, 2(11):6617–6621, 2010. Cited on p. 7.

[9] Inna Vistbakka and Elena Troubitsyna. Towards integrated modelling of dynamic access control with uml and event-b. *arXiv preprint arXiv:1805.05521*, 2018. Cited on p. 7.

[10] Relationship-based access control. https://en.wikipedia.org/wiki/Relationship-based_access_control. Accessed 2025. Cited on p. 7.

[11] Use supabase with flutter. https://supabase.com/docs/guides/getting-started/quickstarts/flutter, 2025. Accessed 2025. Cited on p. 7.

[12] Monterail. Getting started with supabase and flutter: An overview. https://www.monterail.com/blog/getting-started-with-supabase-and-flutter, 2025. Accessed 2025. Cited on p. 7.

[13] Desmond Nelson. Connect supabase to flutter & perform crud operation. https://medium.com/@desmondnelson/connect-supabase-to-flutter-crud-operation-5683b965952f, 2022. Accessed 2025. Cited on p. 7.

[14] supabase_flutter package. https://pub.dev/packages/supabase_flutter, 2025. Accessed 2025. Cited on p. 7.