A close-up of a logo

Description automatically generated

**School of Computing and Digital Technologies**

**Software Architecture and Design**

**(55-608809)**

**Software Architecture and Design Document**

**Project:**  Complaint Management System (CMS)

|  |  |
| --- | --- |
| **Name** | **ID** |
| **Thomas Grout** | c3034731 |

**Date: 13/10/2025**

Table of Contents

[Introduction 3](#_Toc208957300)

[Overview 3](#_Toc208957301)

[Solution Architecture 4](#_Toc208957302)

[Non-Functional Requirements 4](#_Toc208957303)

[C4 Context Diagram (level 1) 4](#_Toc208957304)

[C4 Container Diagram (level 2) 4](#_Toc208957305)

[Technology Stack 4](#_Toc208957306)

[Solution Design 4](#_Toc208957307)

[User Interface Design 4](#_Toc208957308)

[C4 Component Diagram (level 3) 4](#_Toc208957309)

[C4 Code Diagrams (level 4) 4](#_Toc208957310)

[API endpoints 4](#_Toc208957311)

[Data Design 4](#_Toc208957312)

[Security Considerations 4](#_Toc208957313)

[Appendix 1: Transparency Declaration Statements 5](#_Toc208957314)

[Appendix 2: Refined User Stories 5](#_Toc208957315)

[Appendix 3: Use Cases 5](#_Toc208957316)

[Appendix 4: Architecture Decision Records 5](#_Toc208957317)

[Appendix 5: Incorporation of Formative Feedback 6](#_Toc208957318)

[Appendix 6: 7](#_Toc208957319)

[References 8](#_Toc208957320)

# Introduction

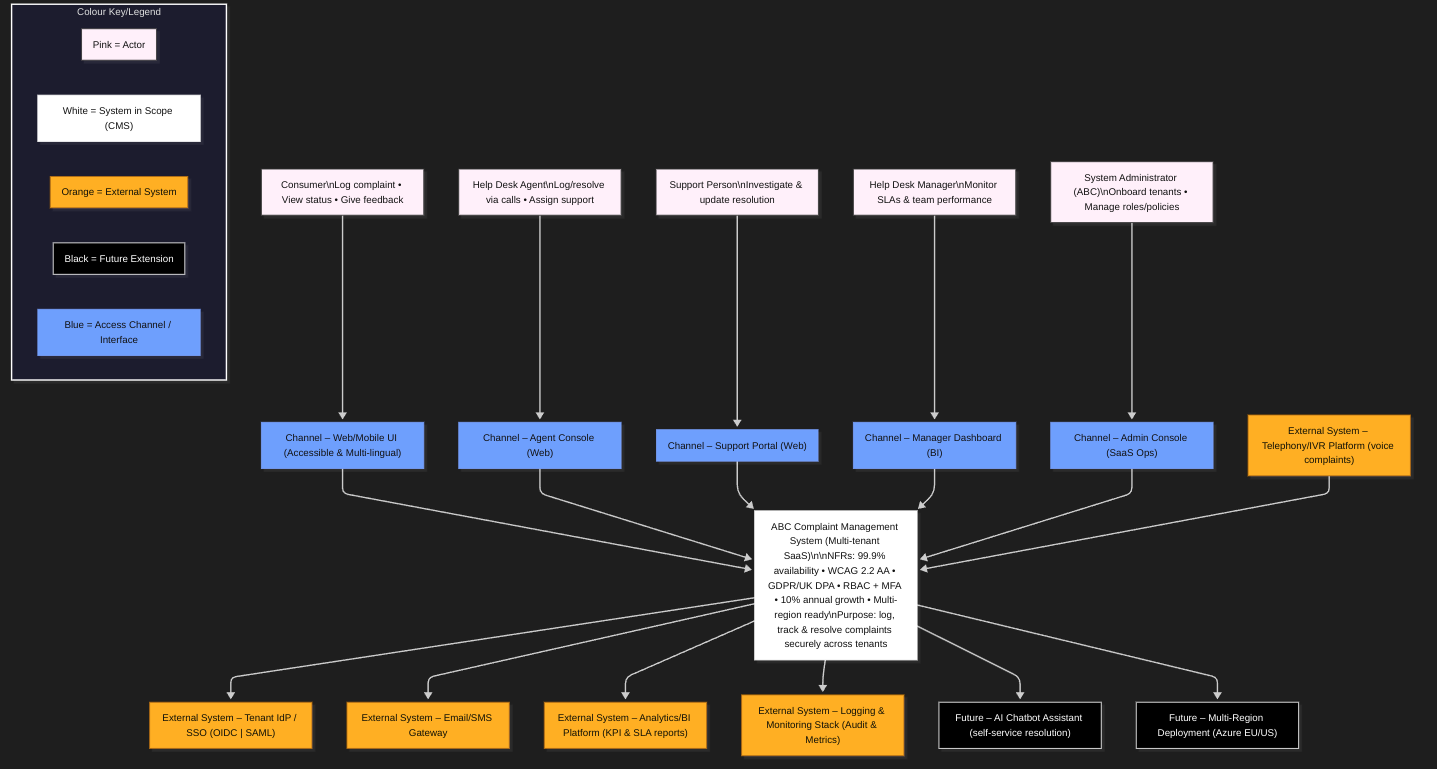
## Overview

# Solution Architecture

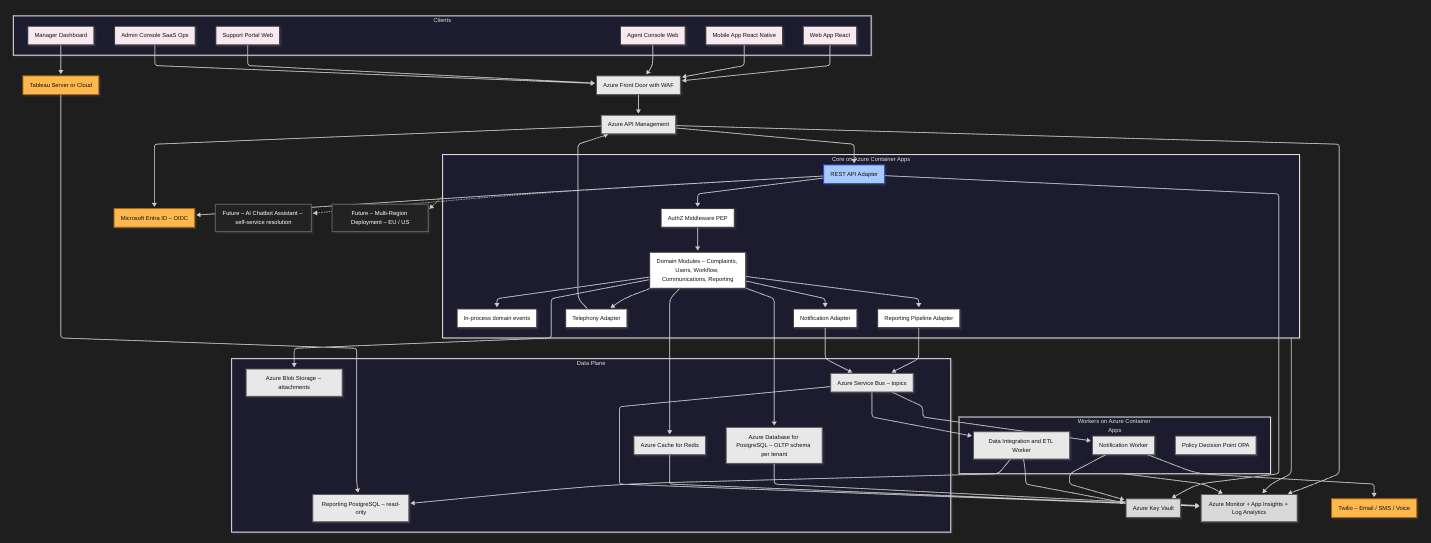
## Non-Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Theme** | **Description** | **Priority (MoSCoW)** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## C4 Context Diagram (level 1)



## C4 Container Diagram (level 2)



This container diagram visualises the modular-monolith CMS core deployed on Azure Container Apps with clear separation of concerns. The API Gateway enforces tenant context and OIDC security before traffic enters the core, which exposes ports (REST, Events, Adapters) consistent with the hexagonal style. The data plane isolates persistence (PostgreSQL schema-per-tenant + RLS, Redis cache, Blob Storage) from domain logic, while workers handle asynchronous responsibilities via Service Bus topics. External integrations—Twilio for notifications, Tableau for analytics, Key Vault for secrets—sit outside the trust boundary to maintain least-privilege access. This design balances simplicity and scalability: a cohesive monolith avoids microservice overheads yet remains evolvable through its well-defined ports, satisfying the SAAD brief’s demand for contemporary, critically justified architecture.

## Technology Stack

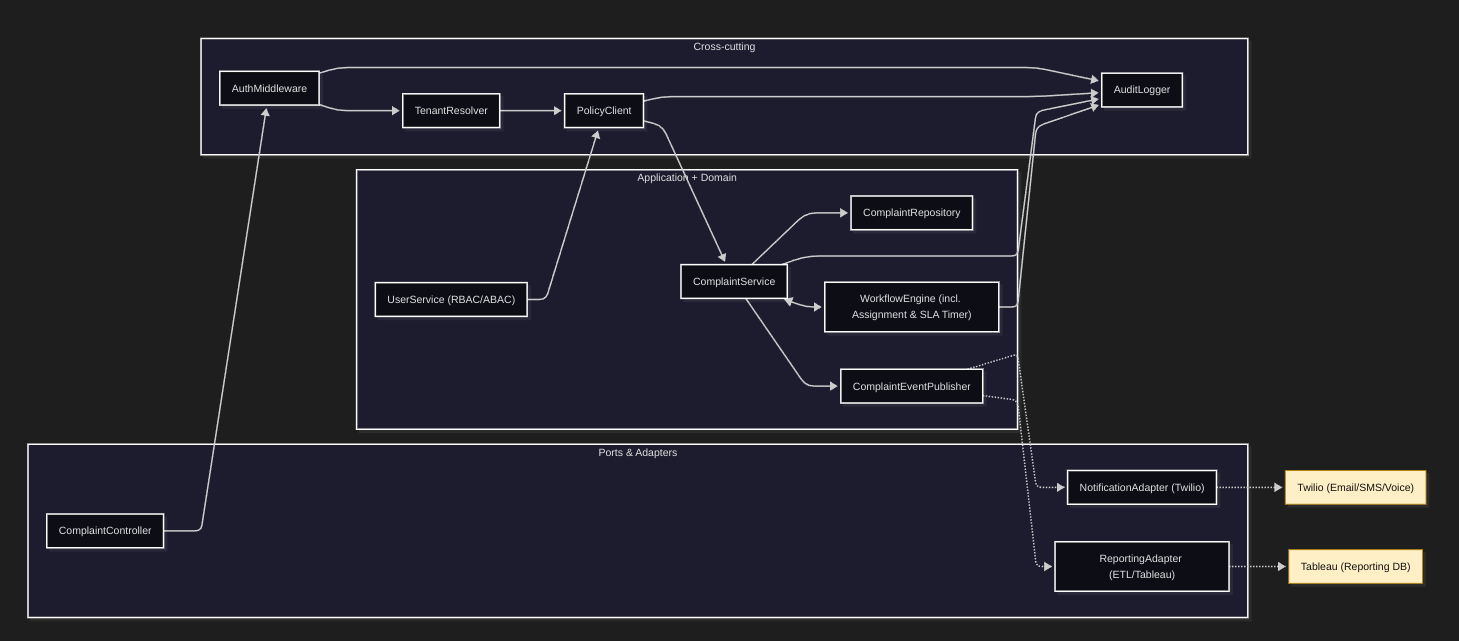
A brief list of the different technologies to be employed in the project.

# Solution Design

## User Interface Design

In this section you should include Wireframes. They should be consistent with the user stories and the non-functional requirements (especially usability requirements)

## C4 Component Diagram (level 3)



This component diagram provides a detailed view of the CMS Core container, revealing the internal structure of the modular monolith based on a Hexagonal (Ports and Adapters) architecture. Controllers within the adapter layer process RESTful requests and delegate to domain services, which coordinate repositories and the workflow engine synchronously while publishing asynchronous domain events through BullMQ. Cross-cutting middleware—**AuthMiddleware**, **TenantResolver**, **PolicyClient**, and **AuditLogger**—enforces multi-tenant security, policy-based access control, and audit traceability. Outbound adapters, including the **NotificationAdapter** (Twilio) and **ReportingAdapter** (Tableau), decouple integrations to support horizontal scalability and maintain isolation of external dependencies. This design realises the case-study use cases for complaint logging, assignment, and resolution, while directly addressing key non-functional requirements of **security**, **scalability**, and **reliability**, in accordance with the assessment brief’s expectations for a coherent, pattern-aligned Level-3 model.

## C4 Code Diagrams (level 4)

* Structural diagram
* Behavioural diagram

## API endpoints

Document your APIs here if your design is service oriented. If your design is monolithic ignore this.

## Data Design

Include diagrams and description of the project data design.

## Security Considerations

Discuss security measures considered in your design; this may include authentication (SSO, multi factor authentication etc.), authorisation (role-based access) , encryption ( data level) etc.

# Appendix 1: Transparency Declaration Statements

If you take help of generative AI, please declare it here.

# Appendix 2: Refined User Stories

If you feel appropriate you may want to refine the user stories provided in the case study. You may take help of generative AI. If you take help, please acknowledge that. You can also refer to SHUDev Process template to refine user stories

# Appendix 3: Use Cases

Identify use cases from the information provided in the case study. You may take help of generative AI. If you take help, please acknowledge that.

# Appendix 4: Architecture Decision Records

Include one key ADR example here.

Add Link to GitHub repository that includes all project ADRs.

## 

# Appendix 5: Incorporation of Formative Feedback

Write here about evaluation of Formative Feedback and Actions Taken in Response to Feedback.

Table 1 - Weekly Feedback

|  |  |  |
| --- | --- | --- |
| **Week #** | **Feedback** | **Response** |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |
| **4** |  |  |
| **5** |  |  |
| **6** |  |  |
| **7** |  |  |
| **8** |  |  |
| **9** |  |  |
| **10** |  |  |
| **11** |  |  |
| **12** |  |  |

# Appendix 6:

Use if needed. You can include github link , information needed to augment your architecture or design or anything else you may feel relevant.

# References