**laytest**

**High Level Design Document**

Version: 1.0

Date: April 09, 2025

Author: iskyfinn@outlook.com

Table of Contents

Right-click to update table of contents.

# Introduction

## Purpose

This High Level Design Document (HLDD) provides an architecture overview for laytest. It describes the system components, data flow, security mechanisms, and deployment architecture.

## Scope

This document covers the high-level architecture design including system components, interfaces, data model, security considerations, and deployment architecture for laytest.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term/Acronym** | **Definition** |
| HLDD | High Level Design Document |
| API | Application Programming Interface |
| DB | Database |

# System Overview

I need an aiml solution to work on residential houses as an electrician

## System Context

This section describes the system context and its interactions with external systems and users.

## System Functions

## User Roles

|  |  |
| --- | --- |
| **Role** | **Description** |
| Administrator | System administrator with full access permissions |
| User | Regular user with standard access permissions |

# Architecture Overview

The system uses a modern, scalable architecture designed for reliability and maintainability. It consists of multiple components that work together to deliver the system functionality.

## Architecture Principles

* Security: The system implements defense in depth security measures
* Scalability: The system is designed to scale horizontally to meet demand
* Reliability: The system is fault-tolerant and designed for high availability

## Architecture Diagram

The following diagram illustrates the high-level architecture of the system:

[Architecture Diagram Placeholder]

## Component Descriptions

### Frontend

User interface that provides access to system functionality

### API Layer

RESTful API that processes requests and manages business logic

### Database

Data storage for application information

# Technology Stack

The system is built using modern technologies chosen for their reliability, performance, and developer productivity.

## Frontend

|  |  |  |
| --- | --- | --- |
| **Technology** | **Version** | **Purpose** |
| React | 17.x | UI framework |

## Backend

|  |  |  |
| --- | --- | --- |
| **Technology** | **Version** | **Purpose** |
| Python | 3.9+ | Programming language |
| Flask | 2.x | Web framework |

## Database

|  |  |  |
| --- | --- | --- |
| **Technology** | **Version** | **Purpose** |
| PostgreSQL | 13.x | Primary database |

# Data Architecture

The system uses a structured approach to data management with appropriate storage solutions for different data types and access patterns.

## Data Stores

### Primary Database

**Type:** Relational **Purpose:** Main application data storage

### Cache

**Type:** In-Memory **Purpose:** Temporary data caching for performance

## Data Models

The following data models are used in the system:

[Data Model Diagram Placeholder]

## Data Flows

The following diagram illustrates the data flows in the system:

[Data Flow Diagram Placeholder]

# Security Architecture

The system implements a defense-in-depth approach to security with multiple protection layers and follows security best practices.

## Authentication and Authorization

The system uses JWT-based authentication with role-based access control (RBAC) to manage user permissions.

## Data Protection

Sensitive data is encrypted in transit using TLS and at rest using AES-256 encryption.

## Network Security

The system uses network segmentation, firewalls, and intrusion detection systems to protect against network-based attacks.

## Compliance

The system is designed to comply with relevant regulatory requirements.

# Deployment Architecture

The system is deployed using a cloud-native approach with containerization for consistency across environments.

## Deployment Diagram

The following diagram illustrates the deployment architecture of the system:

[Deployment Diagram Placeholder]

## Environments

### Development

Used for development and unit testing

### Staging

Used for integration testing and UAT

### Production

Used for the live system

## CI/CD Pipeline

The system uses a CI/CD pipeline for automated testing and deployment.

# Operational Considerations

This section describes the operational considerations for the system.

## Monitoring and Logging

The system implements monitoring and logging mechanisms to ensure operational visibility.

## Backup and Recovery

The system implements backup and recovery mechanisms to ensure data durability.

## Scaling and Performance

The system is designed to scale to handle increasing load and maintain performance.

## Disaster Recovery

The system implements disaster recovery measures to ensure business continuity.

# Appendices

## References

* Reference 1
* Reference 2
* Reference 3

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description** |
| 1.0 | 2025-04-09 | Architecture Team | Initial version |