Leave and Absence Registration System (LARS)

Software Design

[[1]](#endnote-1)AI Free™

**Customer:** GeoProfs B.V.

**Document type:** Software Design Document (SDD)

**Author:** Sintayu De Kuiper Floris van Drunen Odin Vos

**Student number:** 1201903 1203212 1197607

**Class:** TIA4V3a

**Program:** Software Developer (25998), MBO 4

**Institution** ROC Nijmegen

**Date created:** 3 oktober 2024

**Date modified:** 9 oktober 2024

Contents

[Introduction 3](#_Toc179366564)

[Purpose of the Document 3](#_Toc179366565)

[Scope 3](#_Toc179366566)

[Audience 3](#_Toc179366567)

[Referenced Documents 3](#_Toc179366568)

[Definitions and Acronyms 3](#_Toc179366569)

[System Overview 4](#_Toc179366570)

[System Description 4](#_Toc179366571)

[High-Level Architecture 4](#_Toc179366572)

[Back-End Design 5](#_Toc179366573)

[Architectural Design 5](#_Toc179366574)

[Technologies and Frameworks 5](#_Toc179366575)

[Service Layers 5](#_Toc179366576)

[API Design Principles 5](#_Toc179366577)

[Scalability Considerations 5](#_Toc179366578)

[Detailed Design 5](#_Toc179366579)

[API Endpoint Specifications 5](#_Toc179366580)

[Business Logic Implementation 5](#_Toc179366581)

[Authentication and Authorization 6](#_Toc179366582)

[Security Measures 6](#_Toc179366583)

[Front-End Design 7](#_Toc179366584)

[Architectural Design 7](#_Toc179366585)

[Technologies and Frameworks 7](#_Toc179366586)

[Component Structure 7](#_Toc179366587)

[Data Flow 7](#_Toc179366588)

[API Integration 7](#_Toc179366589)

[Detailed Design 7](#_Toc179366590)

[User Interface Design 7](#_Toc179366591)

[Component Details 7](#_Toc179366592)

[Routing and Navigation 8](#_Toc179366593)

[Form Handling and Validation 8](#_Toc179366594)

[Security Measures 8](#_Toc179366595)

[Database Design 9](#_Toc179366596)

[Architectural Design 9](#_Toc179366597)

[Database Choice 9](#_Toc179366598)

[Data Storage Architecture 9](#_Toc179366599)

[Detailed Design 9](#_Toc179366600)

[Schema Definition 9](#_Toc179366601)

[Data Access Layer 9](#_Toc179366602)

[Security Measures 9](#_Toc179366603)

[Security Design 10](#_Toc179366604)

[Overview of Identified Threats 10](#_Toc179366605)

[Mitigation Strategies 10](#_Toc179366606)

[Security Policies and Compliance 10](#_Toc179366607)

[Integration Design 11](#_Toc179366608)

[API Contracts 11](#_Toc179366609)

[Data Formats 11](#_Toc179366610)

[Versioning 11](#_Toc179366611)

[Testing Strategy 12](#_Toc179366612)

[Front-End Testing 12](#_Toc179366613)

[Back-End Testing 12](#_Toc179366614)

[Database Testing 12](#_Toc179366615)

[Appendices 13](#_Toc179366616)

[References 13](#_Toc179366617)

# **Introduction**

## **Purpose of the Document**

this document provides a detailed design for implementing the GeoProfs system, guiding developers in translating the requirements from the SRS into a secure and functional system.

## **Scope**

System architecture, detailed design of front-end, back-end, and database components, as well as security measures.

Addresses how to implement the requirements specified in the SRS, with particular attention to security considerations.

Detailed requirements are available in the SRS, which is referenced in this document.

## **Audience**

* Developers (front- & back-end, API, database)
* Product-owner

## **Referenced Documents**

* [Software Requirements Specification (SRS)](https://rocnijmegen2-my.sharepoint.com/personal/1201903_student_roc-nijmegen_nl/Documents/Jaar%203/GeoProfs/untracked/SRS.Personal.docx)
* **Security Threat Analysis Document**
* **Coding Standards and Guidelines**

## 

## **Definitions and Acronyms**

# **System Overview**

## **System Description**

* **Explanation**: Provide a high-level description of the system.
* **Content**:
  + Briefly describe the GeoProfs system, its objectives, and main functionalities.
  + Reference the SRS for detailed functional and non-functional requirements.

The custom made GeoProfs Leave and Absence Registration System (LARS) will allow employees to request off days of different types (sick, vacation, personal).

## **High-Level Architecture**

* **Explanation**: Present the overall system architecture.
* **Content**:
  + **Architecture Diagram**: Include diagrams showing the main components (front-end, back-end, database, third-party services) and their interactions.
  + **Component Interaction**: Explain how components communicate, emphasizing the use of APIs for front-end and back-end communication.

# **Back-End Design**

#### Architectural Design

##### Technologies and Frameworks

* **Languages and Frameworks: Laravel 11, PHP**
* **Database Interaction**: Laravel Eloquent

##### Service Layers

* **Explanation**: Describe the layers in the back-end.
* **Content**:
  + **Controllers**: Handling HTTP requests.
  + **Services**: Business logic implementation.
  + **Repositories/DAOs**: Data access logic.

##### API Design Principles

* **Explanation**: Outline principles for API development.
* **Content**:
  + **RESTful Principles**: Statelessness, resource naming.
  + **Versioning**: Strategies to manage API versions.

##### Scalability Considerations

* **Explanation**: Discuss how the back-end scales.
* **Content**:
  + **Load Balancing**: Strategies for distributing traffic.
  + **Caching Mechanisms**: Use of Redis, Memcached.

#### Detailed Design

##### API Endpoint Specifications

* **Explanation**: Provide detailed API information.
* **Content**:
  + **Endpoint List**: URLs, methods, parameters.
  + **Request/Response Models**: JSON schemas, data types.
  + **Error Responses**: Standardized error messages and codes.

##### Business Logic Implementation

* **Explanation**: Detail the core functionality.
* **Content**:
  + **Algorithms and Workflows**: Steps for processing data.
  + **Edge Cases**: Handling of exceptional scenarios.

##### Authentication and Authorization

* **Explanation**: Define security mechanisms.
* **Content**:
  + **Authentication Methods**: JWT, OAuth2 flows.
  + **Authorization Checks**: Role-based access control (RBAC), permissions.

##### Security Measures

* **Explanation**: Outline back-end security practices.
* **Content**:
  + **Input Validation**: Libraries and methods used.
  + **SQL Injection Prevention**: Use of parameterized queries.
  + **Error Handling**: Avoiding information leakage.

# Front-End Design

#### Architectural Design

##### Technologies and Frameworks

* **Explanation**: Specify the tools to be used.
* **Content**:
  + **Frameworks**: E.g., React.js, Angular, or Vue.js.
  + **Languages**: HTML5, CSS3, JavaScript/TypeScript.
  + **State Management**: E.g., Redux, Vuex.

##### Component Structure

* **Explanation**: Describe the modular structure.
* **Content**:
  + **Component Hierarchy Diagrams**: Show parent-child relationships.
  + **Reusability**: Strategies for creating reusable components.

##### Data Flow

* **Explanation**: Explain how data moves within the front-end.
* **Content**:
  + **State Management Patterns**: Unidirectional data flow.
  + **Data Binding**: Methods used for binding data to UI elements.

##### API Integration

* **Explanation**: Detail how the front-end communicates with the back-end.
* **Content**:
  + **HTTP Clients**: Use of fetch API, Axios, or other libraries.
  + **Error Handling**: Strategies for handling API errors gracefully.

## **Detailed Design**

##### User Interface Design

* **Explanation**: Provide detailed UI designs.
* **Content**:
  + **Wireframes and Mockups**: Visual representations of all screens.
  + **Style Guides**: Fonts, colors, and design principles.
  + **Accessibility Considerations**: Compliance with WCAG guidelines.

##### Component Details

* **Explanation**: Dive into specific components.
* **Content**:
  + **Component Specifications**: Props, state, and lifecycle methods.
  + **Interaction Patterns**: How components interact with each other.

##### Routing and Navigation

* **Explanation**: Define the navigation structure.
* **Content**:
  + **Routing Libraries**: E.g., React Router.
  + **URL Structure**: Mapping between URLs and components.

##### Form Handling and Validation

* **Explanation**: Explain how forms are managed.
* **Content**:
  + **Validation Libraries**: E.g., Formik, Yup.
  + **Input Sanitization**: Preventing injection attacks at the client side.

##### Security Measures

* **Explanation**: Outline front-end security practices.
* **Content**:
  + **Content Security Policy (CSP)**: Policies to prevent XSS attacks.
  + **Secure Storage**: Handling of tokens and sensitive data.
  + **Third-Party Scripts**: Managing and auditing external scripts.

# Database Design

#### Architectural Design

##### Database Choice

* **Explanation**: Justify the selected database technology.
* **Content**:
  + **Type of Database**: Relational (e.g., PostgreSQL) or NoSQL (e.g., MongoDB).
  + **Reasons for Choice**: Performance, scalability, data consistency.

##### Data Storage Architecture

* **Explanation**: Describe how data is stored and accessed.
* **Content**:
  + **Normalization**: Levels of normalization used.
  + **Indexing Strategies**: To optimize query performance.

#### Detailed Design

##### Schema Definition

* **Explanation**: Provide detailed schema designs.
* **Content**:
  + **Entity-Relationship Diagrams (ERDs)**: Visual representation of tables and relationships.
  + **Table Definitions**: Columns, data types, constraints.

##### Data Access Layer

* **Explanation**: Explain how the application interacts with the database.
* **Content**:
  + **ORM Configuration**: Models, mappings.
  + **Query Methods**: Best practices for querying data.

##### Security Measures

* **Explanation**: Outline database security practices.
* **Content**:
  + **Access Control**: Database user permissions.
  + **Encryption**: Data at rest encryption methods.
  + **Backup and Recovery**: Strategies for data protection.

# Security Design

#### Overview of Identified Threats

* **Explanation**: Summarize the security threats.
* **Content**: List threats such as MitM attacks, SQL injection, CSRF, etc.

#### Mitigation Strategies

* **Explanation**: Detail how each threat is addressed.
* **Content**:
  + **Threat**: Description and potential impact.
  + **Mitigation**: Specific measures, including implementation details and code examples.

#### Security Policies and Compliance

* **Explanation**: Address adherence to security standards.
* **Content**:
  + **OWASP Top Ten**: How the design mitigates these risks.
  + **Regulatory Compliance**: GDPR, CCPA considerations.

# Integration Design

#### API Contracts

* **Explanation**: Define the agreements between front-end and back-end.
* **Content**:
  + **Data Models**: Consistent models used across components.
  + **Versioning and Compatibility**: Ensuring smooth integration over time.

#### Data Formats

* **Explanation**: Specify data exchange formats.
* **Content**:
  + **JSON Schemas**: For request and response bodies.
  + **Serialization/Deserialization**: Handling of data encoding.

#### Versioning

* **Explanation**: Plan for API evolution.
* **Content**:
  + **Versioning Strategy**: URL-based, header-based approaches.
  + **Deprecation Policies**: How and when old versions are phased out.

# Testing Strategy

#### Front-End Testing

* **Explanation**: Define front-end testing practices.
* **Content**:
  + **Unit Tests**: Testing individual components.
  + **Integration Tests**: Testing component interactions.
  + **End-to-End Tests**: Simulating user interactions.

#### Back-End Testing

* **Explanation**: Define back-end testing practices.
* **Content**:
  + **Unit Tests**: Testing services and controllers.
  + **Integration Tests**: Database interactions.
  + **API Testing**: Validation of API endpoints.

#### Database Testing

* **Explanation**: Ensure data integrity and performance.
* **Content**:
  + **Migration Tests**: Validating schema changes.
  + **Performance Tests**: Query optimization.

# Appendices

## **References**

* **Explanation**: Cite all referenced materials.
* **Content**:
  + **Documents**: SRS, coding standards, security policies.
  + **Web Resources**: Links to frameworks, libraries, and tools.
  + **Standards and Regulations**: OWASP Top Ten, GDPR guidelines.

1. No content, text or material in this document has been generated by AI (except this footnote). While AI may have been used for inspiration or gathering information, all content is fully original and created without AI tools. [↑](#endnote-ref-1)