Design Document

**Teachers:** Michiel Koehorst

**Student Name:** Omar Abou Dehn

**Student Number:** 3560813

**Interface Git Repository:** https://git.fhict.nl/I407846/simac\_group2

Contents

[Architecture: 3](#_Toc105783142)

[Three Tier Architecture: 3](#_Toc105783143)

[SOLID Design Principle: 4](#_Toc105783144)

[Entity Relational Diagram: 5](#_Toc105783145)

[Backend Framework: 6](#_Toc105783146)

[Frontend Framework: 7](#_Toc105783147)

[Database Framework: 7](#_Toc105783148)

# Architecture:

## Three Tier Architecture:

The three-tier architecture will be used to build the web-app, which consists of a presentation layer built using React framework, a logic layer built using SpringBoot, a persistence layer using MySQL.

Some of the advantages of using it are that it can be developed fast as multiple teams can work on different layers, and since the layers are separated, they can be scaled independently from one another, moreover the data coming from the frontend can be processed and validated in the backend before being inserted to the database which increases security.

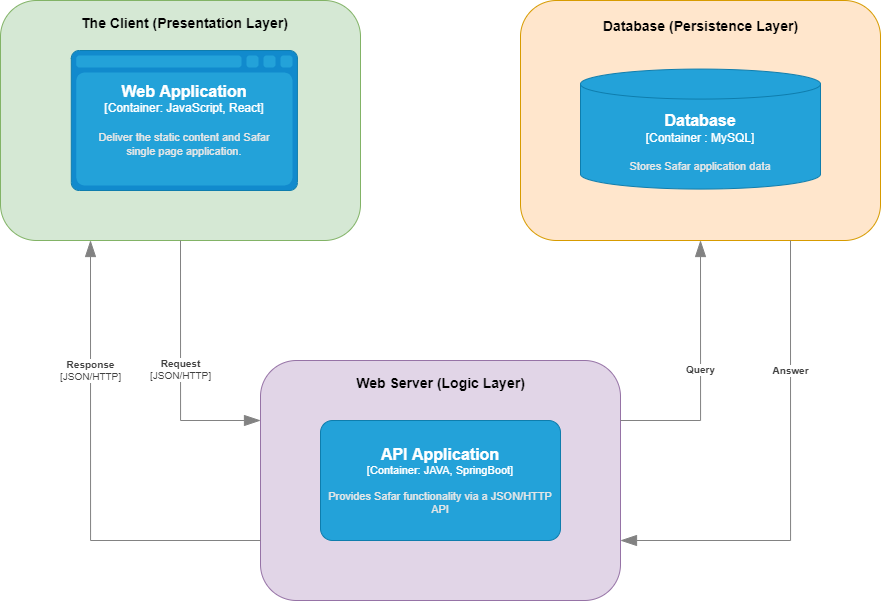


Figure 1 Three Tier Architecture

## SOLID Design Principle:

SOLID principles are a set of five principles that ensure good Object-Oriented Design.  
Those principles are:

1. Single Responsibility Principle: A class should have one and only one reason to change, meaning that a class should have only one job.
2. Open Closed Principle: Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.  
   The idea behind this principle, is to ensure that the functions/classes have scope for extension in the future. New features can be added to it, without introducing new bugs. You will add new features by extending or writing new code without modifying old code.
3. Liskov Substitution Principle: functions that use pointers to base classes must be able to use objects of derived classes without knowing it.
4. Interface Segregation Principle: A client should never be forced to implement an interface that it does not use, or clients should not be forced to depend on methods they do not use.
5. Dependency Inversion Principle: Entities must depend on abstractions not on concretions. It states that the high-level module must not depend on the low-level module, but they should depend on abstractions.

During this project, the above-mentioned principles will be followed and applied when relevant.

# Entity Relational Diagram:

Diagram

Description automatically generated

# Backend Framework:

Spring Boot is an open-source Java-based framework used to build stand-alone and production ready spring applications.  
It allows developers to get started with minimum configurations without the need for an entire Spring configuration setup.

Some of the benefits of using Spring Boot are:

* Flexibility: Its flexible and comprehensive set of extensions and third-party libraries let developers build almost any application imaginable with a wide-ranging set of features and functionality.
* Productivity: It offers annotation-based applications, and manual configurations are not needed as everything is configured automatically.
* Security: Spring deals quickly with security issues and reported vulnerabilities, and it contains a highly customized and powerful security framework called Spring Security, which focuses on providing authentication and authorization to Java applications.
* Popularity and supportiveness: Spring is the most famous Java framework, and it has a large global community which subsequently provides a lot of support for spring developers.

For the above-mentioned reasons, Spring will be used to build the backend.

# Frontend Framework:

React is a JavaScript library for building single-page web applications, it is one of the best most famous JavaScript library/framework. Due to its modular structure and reusable components, it is flexible, productive, and easy to learn and maintain. React also uses the concept of a virtual DOM to minimize the performance cost of re-rendering a webpage because the actual DOM is expensive to manipulate, in other words it speeds up the UI render time.

For the above-mentioned reasons React will be used to build the interface.

# Database Framework:

|  |  |  |
| --- | --- | --- |
|  | **JDBC** | **JPA** |
| **Abstraction Level** | Low level standard for interaction with databases | High level standard for interaction with databases |
| **Usage** | JDBC allows to do more things with the Database directly which gives more flexibility | JPA allows to use an object model in the application which requires less code |
| **Migrating Database** | Due to its low abstraction, a lot of changes are required when changing the database | Due to its high abstraction, the database can be switched by changing some configurations |