



SOFTWARE DESIGN SPECIFICATION

JAVA  
  
IAC ‘COMMUNITY AMBASSADOR ONBOARDING PROCESS’

|  |  |  |  |
| --- | --- | --- | --- |
| **Created By:** | GUGULOTH RAJKUMAR | **Approved By:** | Harshada Topale |
| **Created On:** | 09-04-2025 | **Approved On:** | DD-MMM-YYYY |

Page left blank intentionally

**INDEX**

[**1** **PURPOSE** 2](#_Toc142418236)

[**2** **PROJECT SCOPE** 2](#_Toc142418237)

[**3** **SYSTEM OVERVIEW** 2](#_Toc142418238)

[**4** **DESIGN CONSIDERATIONS** 2](#_Toc142418239)

[4.1 Requirements 3](#_Toc142418240)

[4.2 Assumptions 3](#_Toc142418241)

[4.3 Dependencies 3](#_Toc142418242)

[**5** **SYSTEM ARCHITECTURE** 3](#_Toc142418243)

[5.1 Architectural Strategies 4](#_Toc142418244)

[5.2 Structure & Relationships 4](#_Toc142418245)

[**6** **DETAILED DESCRIPTION OF COMPONENTS** 4](#_Toc142418246)

[**7** **INTEGRATION** 5](#_Toc142418247)

[**8** **APPENDICES** 1](#_Toc142418248)

[8.1 Appendix A – Detailed Description of Components 1](#_Toc142418249)

**General Instructions for using the Live Project POC Document**

* This template and the subsequent document created using this template is a confidential document and is the intellectual property of Cloud Counselage Pvt. Ltd. Circulating it outside of the organisation without the consent of Cloud Counselage Pvt. Ltd. is the breach of company policies and will lead to legal actions
* The Design Specification of a software forms the basis of development of software
* The **text between inequality (< >) is to be replaced** by relevant text
* Please **remove the yellow highlight on the Text** between the inequality (< >). This is done to help you notice the text to be changed/replaced
* The text in *italics* highlighted in grey is just for reference and should be removed after adding the relevant text

# **PURPOSE**

This document is created based on the requirement specification document. The purpose of this Software Design Specification (SDS) Document is to break down the project into components to describe in detail what the purpose of each component is and how it will be implemented. The SDS will also serve as a tool for verification and validation of the final product.

# **PROJECT SCOPE**

The scope of the IAC ‘Community Ambassador Onboarding Process’ includes its distinct features, its benefits, and its limitations. The system's distinct features allow it to “develop a Java application to automate the CA Onboarding workflows and their respective Performance Dashboards (linked to their respective performances @ UTMs, etc.) Once the CA submits the New Joinee Form, they receive a welcome email with the UTM link for the IAC web page <https://www.industryacademiacommunity.com/>   
1. Create a UTM link as soon as the submission is received in the new Joinee Form.   
2. Share this UTM link to the email id received in the 'New Joinee Form' as part of the Welcome email.” by using < Questionnaire, Requirement Elicitation Meeting Notes, SRS (Software Requirement Specification) Document, Design Specification Template, STS (Spring Tool Suite), HTML, CSS, Bootstrap, JavaScript, Oracle Database (SQL Developer), Postman, Maven, Spring Security, Java MailSender (Spring), JUnit, Traceability Matrix Template etc..>. The system enables the user to register Community Ambassadors, track their performance, and send confirmation emails while allowing admins to monitor signups and export performance reports in PDF/Excel formats.

# **SYSTEM OVERVIEW**

This section will provide an outline of the various components and subsystems of “IAC COMMUNITY AMBASSADOR ONBOARDING PROCESS”

**Model:**

* The model represents the data structure of the system. It includes entities such as Community Ambassador, Signup, and Performance Metrics.
* **Key Components**:
  + *Community Ambassador*: Represents the ambassadors, their signups, and related information.
  + *Signup*: Tracks the signup status and other associated data.
  + *Performance Metrics*: Contains data related to the performance of ambassadors, including metrics like signups and engagement.

**View:**

* The view is responsible for displaying the data to the users. It’s built using Angular, HTML, and CSS.
* It allows the system's users to interact with the data, such as viewing the dashboard, registering for the platform, and managing signups.

**Controller:**

* The controller acts as an intermediary between the view and the model. It processes user inputs, calls the appropriate methods from the model, and updates the view accordingly.
* **Key Components**:
  + *Admin Controller*: Handles requests related to admin actions, such as viewing ambassador performance.
  + *Signup Controller*: Manages requests related to ambassador registration and signup tracking.

**Flow of the Application:**

1. The user (admin or ambassador) interacts with the **View** (frontend).
2. The **Controller** receives the input and calls the appropriate logic in the **Model** (backend).
3. The **Model** processes the data, stores it in the **Database**, and sends the required response back to the **Controller**.
4. The **Controller** updates the **View**, displaying the updated information to the user.

# **DESIGN CONSIDERATIONS**

This section describes requirements, assumptions and dependencies to be addressed to devise a complete design solution.

## Requirements

<Add requirements as identified in the Software Requirement (SRS) document> The list of components

## Assumptions

<Add assumptions as listed in the Software Requirement (SRS) document> The list of components

## Dependencies

<Add assumptions as listed in the Software Requirement (SRS) document> The list of components

# **SYSTEM ARCHITECTURE**

The software system architecture refers to the logical organization of a distributed system into software components. It defines how components of a software system are assembled, their relationship and communication between them. It serves as a blueprint for software application and development basis for developer team. An effective architecture serves as the conceptual glue that holds every phase of the project together for all of its stakeholders, enabling agility, time and cost savings, and early identification of design risks.

The Software architecture:

* Defines structure of a system
* Defines behaviour of a system
* Defines component relationship
* Defines communication structure
* Balances stakeholder’s needs
* Influences team structure
* Focuses on significant elements
* Captures early design decisions

Below some important characteristics which are commonly considered are explained.

**Operational Architecture Characteristics:**

* Availability
* Performance
* Reliability
* Low fault tolerance
* Scalability

**Structural Architecture Characteristics:**

* Configurability
* Extensibility
* Supportability
* Portability
* Maintainability

**Cross-Cutting Architecture Characteristics:**

* Accessibility
* Security
* Usability
* Privacy
* Feasibility

## Architectural Strategies

## Frontend (View):

## Built using Angular, HTML, CSS, and JavaScript.

## Displays user interfaces for registration, dashboard, and performance metrics.

## Backend (Controller):

## Built with Spring Boot and Java.

## Handles business logic, requests, and manages the flow of data.

## Database (Model):

## Uses Oracle DB to store user, signup, and performance data.

## Ensures data integrity and retrieval.

## Email Service:

## Sends registration and performance emails using SMTP integration.

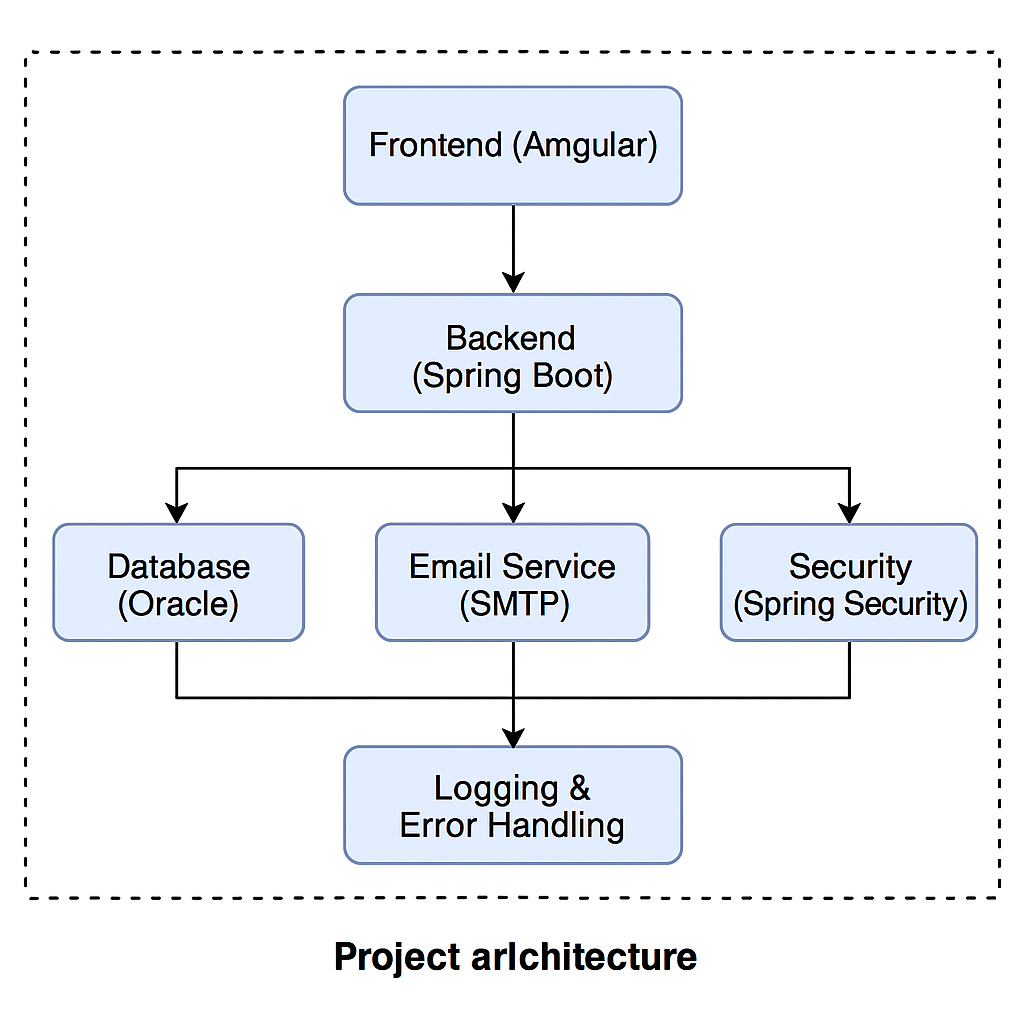
## Security:

## Utilizes Spring Security to manage user authentication and authorization.

## Logging and Error Handling:

## Centralized logging and exception handling for smooth monitoring and debugging.

## Structure & Relationships



# **DETAILED DESCRIPTION OF COMPONENTS**

For detailed description of the components, please refer **Appendix A – Detailed Description of Components**

The below template will be used to specify the details of all the components

**Table 1: Detailed Design Specification Template**

|  |  |
| --- | --- |
| **Identification** | The unique name for the component and the location of the component in the system. |
| **Type** | A module, a subprogram, a form, a data file, a control procedure, a class, etc. |
| **Purpose** | Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS - but are implied or adjunct to formally stated SDS requirements. |
| **Subordinates** | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| **Dependencies** | How the component’s function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components. |
| **Interfaces** | Detailed description of all external or internal interfaces as well as of any mechanism for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here. |
| **Resources** | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. |
| **Processing** | A full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| **Data** | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. |

# **INTEGRATIONS**

The system integrates with the following tools and applications:

* **Oracle Database** for storing and retrieving user and signup data.
* **Spring Boot Mail API** for sending confirmation and notification emails.
* **Frontend (HTML/CSS/JavaScript)** via REST APIs to enable user interaction.
* **GitHub** for version control and collaboration.
* **Postman** for API testing and validation.

# **APPENDICES**

## Appendix A – Detailed Description of Components

|  |  |
| --- | --- |
| **Identification** | Community Ambassador Performance Tracking System |
| **Type** | Class/Form/ |
| **Purpose** | |  | | --- | |  |  |  | | --- | | To register ambassadors, track signups via UTM links, and analyze performance | |
| **Subordinates** | Registration Module, Dashboard, Email Service, Admin Reports |
| **Dependencies** | Oracle DB, Mail Server, Spring Boot, Frontend Browser Compatibility |
| **Interfaces** | RESTful APIs between Frontend ↔ Backend ↔ Database |
| **Resources** | |  | | --- | |  |  |  | | --- | | Spring Boot, Oracle, HTML, CSS, JavaScript, GitHub, Postman | |
| **Processing** | Input via forms → Validated → Stored in DB → Reported to Admin |
| **Data** | User details, email, UTM link, signup count, timestamps |

|  |  |
| --- | --- |
| **Identification** | Community Ambassador Performance Tracking System |
| **Type** | Class/Form/ |
| **Purpose** |  |
| **Subordinates** |  |
| **Dependencies** |  |
| **Interfaces** |  |
| **Resources** |  |
| **Processing** |  |
| **Data** |  |