



SOFTWARE DESIGN SPECIFICATION

JAVA  
  
IAC Community Ambassador Onboarding System

|  |  |  |  |
| --- | --- | --- | --- |
| **Created By:** | Rashmi Sahani | **Approved By:** |  |
| **Created On:** | 30-05 -2025 | **Approved On:** |  |

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**

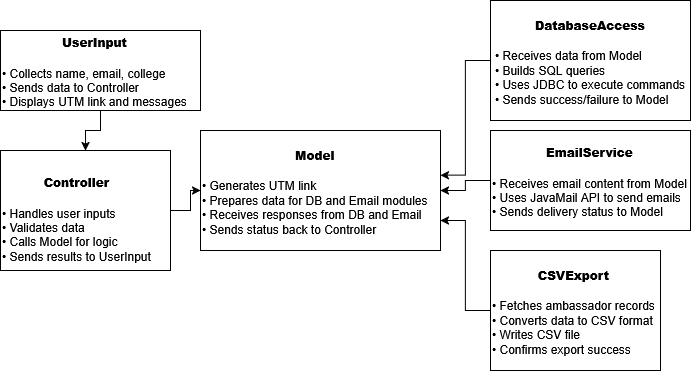
This document is created based on the requirement specification document. The purpose of this Software Design Specification (SDS) Document is to break down the project into components to describe in detail what the purpose of each component is and how it will be implemented. The SDS will also serve as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

# **PROJECT SCOPE**

The scope of the IAC Community Ambassador Onboarding System includes its distinct features, its benefits, and its limitations. The system's distinct features allow it to automate the onboarding workflow by generating unique UTM links, sending personalized welcome emails, and storing ambassador data using Java, JDBC, MySQL, and JavaMail API. The system enables the user to eliminate manual tracking by automatically handling ambassador data entry, link generation, communication, and basic performance tracking — all through a single, lightweight Java console application. While it does not include a graphical interface or real-time analytics in this version, it lays the foundation for scalable future upgrades.

# **SYSTEM OVERVIEW**

This section will provide an outline of the various components and subsystems of IAC Community Ambassador Onboarding System



FOR EXAMPLE: DELETE THIS IMAGE BFORE SUBMITTING

# **DESIGN CONSIDERATIONS**

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

## Requirements

1. **User Input Module**  
   This part collects the name, email, and college of each ambassador through the console. It also checks if the input is valid before moving forward.
2. **UTM Generator**  
   Once the input is received, this module creates a unique UTM link using the ambassador’s name. This link helps track how active they are in spreading awareness.
3. **Database Module**  
   This connects the app to a MySQL database using JDBC. It saves all ambassador details and supports actions like viewing, searching, updating, and deleting records.
4. **Email Module**  
   This part sends a welcome email to the ambassador using the JavaMail API. It uses Gmail’s SMTP server and lets the system know if the email was sent successfully.
5. **Admin Tools**  
   These tools let the user manage all ambassadors — they can view everyone, search by email, make changes, delete entries, and even see how many CAs are registered.
6. **CSV Export Feature**  
   This allows the user to export all the ambassador data into a CSV file that can be used for reports or backups.

## Assumptions

1. **Java and MySQL Setup**  
   It is assumed that Java (JDK 17 or higher) and MySQL are already installed and properly configured on the system where the application will be run.
2. **Console-Based Operation**  
   The user will run the application in a terminal or console environment and be comfortable entering text-based inputs.
3. **Gmail SMTP Configuration**  
   The Gmail account used for sending emails must have “App Passwords” enabled and less secure app access (if required) properly configured.
4. **Stable Internet Connection**  
   Internet access will be available when sending real emails or pushing code to GitHub.
5. **Valid User Inputs**  
   The data entered by the user (such as name, email, and college) is assumed to be genuine and correctly formatted.
6. **File Access**  
   The system will have the required file permissions to create and write the CSV export file.

## Dependencies

1. **Java Development Kit (JDK 17+)**  
   The entire application is written in Java, so it depends on having the correct version of JDK installed and configured on the system.
2. **IntelliJ IDEA (or any Java IDE)**  
   The project is developed and run using IntelliJ IDEA. All code and testing rely on this development environment or a similar IDE.
3. **MySQL Server & MySQL Workbench**  
   The system needs MySQL installed for storing and managing ambassador data. MySQL Workbench is used for easier access and queries.
4. **JDBC Driver (MySQL Connector)**  
   The application requires the JDBC driver to connect Java with MySQL. Without this dependency, database operations will fail.
5. **JavaMail API**  
   The welcome email feature relies on JavaMail. It allows the system to send emails using Gmail's SMTP server.
6. **Gmail App Password**  
   Sending real emails depends on a valid Gmail account with App Password enabled. This setup is critical for email delivery.
7. **Internet Connection**  
   Some features like real email sending and pushing code to GitHub require an active internet connection.
8. **Operating System Support**  
   The system is expected to run on standard Windows, macOS, or Linux environments that support Java and MySQL installations.

# **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

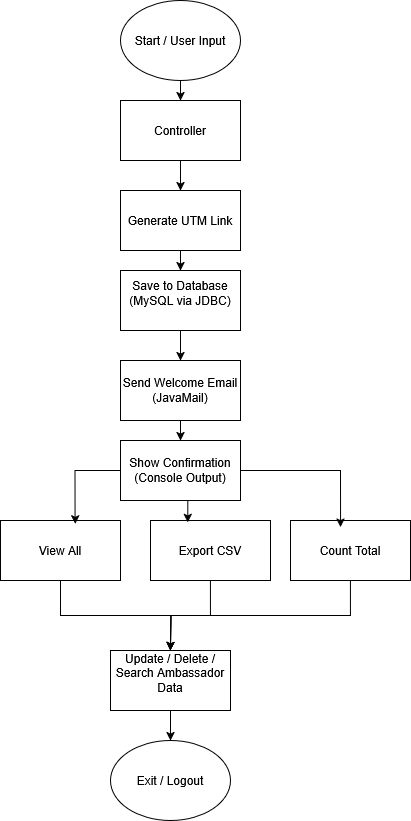
**Cross-Cutting Architecture Characteristics:**

* Accessibility
* Security
* Usability
* Privacy
* Feasibility

## Architectural Strategies

1. **User Interface Layer (Console Input/Output)**  
   Collects user input (name, email, college)  
   Displays UTM link, messages, and admin options
2. **Controller Layer**  
   Validates user input  
   Coordinates between modules (Model, Database, Email, CSV)  
   Handles user flow logic and error handling
3. **UTM Generator Module**  
   Creates a unique UTM link using the user’s name  
   Ensures consistent formatting of the tracking URL
4. **Database Access Layer (JDBC + MySQL)**  
   Stores ambassador data using SQL queries  
   Handles insert, search, update, delete, and view functions
5. **Email Sending Module (JavaMail API)**  
   Sends personalized welcome emails to ambassadors  
   Uses Gmail SMTP server with app password authentication
6. **CSV Export Module**  
    Retrieves all ambassador data from the database  
    Converts the data into .csv format and saves it locally
7. **Admin Functionality Module**  
   – Supports operations like search by email, update details, delete records, and display total count

## 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 IAC Community Ambassador Onboarding System connects with the **JavaMail API** to send welcome emails. It uses Gmail’s SMTP service, and for real email sending, a Gmail App Password needs to be set up in the code. Once it’s configured, the system can automatically email each new ambassador with their UTM link.

The system stores all ambassador data in a **MySQL database**. This includes their name, email, college, and UTM link. The database is local and can be accessed using MySQL Workbench or directly from the code.

To make this connection work, the app uses the **MySQL JDBC Driver**. It helps Java and MySQL talk to each other, so the system can save and read ambassador records.

There’s also a **CSV export feature** in the app. It lets the user download all ambassador data into a .csv file, which can then be opened in Excel or Google Sheets for reporting.

For email to work, the system needs a **stable internet connection**. It connects securely to Gmail using SMTP (port 587 with TLS), and the app is set up to handle failures if something goes wrong during email sending.

Lastly, everything is version-controlled and backed up on **GitHub**. The full project code, updates, and documents are stored there, making it easy to access or submit later.

# **APPENDICES**

## Appendix A – Detailed Description of Components

|  |  |
| --- | --- |
| **Identification** | **CommunityAmbassadorApp** |
| **Type** | Class |
| **Purpose** | Main entry point; controls overall flow of the app |
| **Subordinates** | UTMLinkGenerator, DatabaseModule, EmailService, CSVExporter |
| **Dependencies** | Requires Java 17+, MySQL 8.0, JDBC Driver, JavaMail API |
| **Interfaces** | Console input/output (text-based), method calls to sub-modules |
| **Resources** | |  | | --- | |  |  |  | | --- | | Local MySQL database, Gmail account for email | |
| **Processing** | Handles input → generates UTM → saves to DB → sends email → handles admin tools |
| **Data** | Inputs: name, email, college Outputs: UTM link, confirmation, status messages |

|  |  |
| --- | --- |
| **Identification** | **UTMLinkGenerator** |
| **Type** | Class |
| **Purpose** | Generates a clean UTM link based on the ambassador’s name |
| **Subordinates** | None |
| **Dependencies** | Called by Controller in main app |
| **Interfaces** | |  | | --- | |  |  |  | | --- | | Method returns string UTM link to caller | |
| **Resources** | |  | | --- | |  |  |  | | --- | | None | |
| **Processing** | |  | | --- | |  |  |  | | --- | | Converts name to lowercase, removes spaces, appends to base URL | |
| **Data** | |  | | --- | |  |  |  | | --- | | Input: name Output: formatted UTM link | |

|  |  |
| --- | --- |
| **Identification** | **DatabaseModule** |
| **Type** | Class |
| **Purpose** | Manages ambassador data in MySQL database |
| **Subordinates** | None |
| **Dependencies** | MySQL Server, JDBC (MySQL Connector/J) |
| **Interfaces** | Methods for insert, view, update, delete, count, and search |
| **Resources** | MySQL Workbench / Server |
| **Processing** | Executes SQL queries and returns results |
| **Data** | name, email, college, UTM link |

|  |  |
| --- | --- |
| **Identification** | **EmailService** |
| **Type** | Class |
| **Purpose** | Sends welcome emails using JavaMail API and Gmail SMTP |
| **Subordinates** | None |
| **Dependencies** | JavaMail API, Gmail account, internet connection |
| **Interfaces** | JavaMail SMTP server (smtp.gmail.com, port 587) |
| **Resources** | |  | | --- | |  |  |  | | --- | | Gmail account with app password | |
| **Processing** | Composes message, sets headers, sends via SMTP |
| **Data** | To: email address Body: welcome message + UTM |

|  |  |
| --- | --- |
| **Identification** | **CSVExporter** |
| **Type** | Class |
| **Purpose** | Exports ambassador data from DB to a .csv file |
| **Subordinates** | None |
| **Dependencies** | FileWriter (Java IO), DatabaseModule |
| **Interfaces** | Generates .csv output from DB records |
| **Resources** | |  | | --- | |  |  |  | | --- | | System file storage (local drive) | |
| **Processing** | Loops through DB data and writes rows to CSV |
| **Data** | |  | | --- | |  |  |  | | --- | | All fields: name, email, college, UTM link | |