

Table Of Content

1. INTRODUCTION	6
2. OBJECTIVE	6
3. KEY FEATURES	6
4. DESIGN:	7
4.1: Flow Execution of SWIFTCONNECT:	7
4.2: UI wireframes or pictorial representation:	9
4.2.1: SignUp Page:	9
4.2.2: Login Page:	9
4.2.3: Profile Page:	9
4.2.4: Add new contact Page:	10
4.2.5: Search & Export Page:	10
4.3 Backend:	10
5. TECHNOLOGY STACK.....	11
6. DEVELOPMENT METHODOLOGY	12
7. BENEFITS:	13
8. LIMITATIONS & SOLUTIONS:	14
9. DETAILED PLAN OF WORK	15
10. REFERENCES:	16

1. INTRODUCTION

SwiftConnect is a **Contact Management Application** which offers a robust platform for securely storing and managing contact information. Built with user convenience in mind, the system provides a streamlined interface for adding, editing, and searching for contacts.

This application is designed to efficiently store, organize, and manage contact information. It leverages modern technologies such as Spring Boot, Spring Framework, and Thymeleaf to provide a user-friendly interface. Key features include secure login, contact management, and email integration. While primarily focused on contact management, the application's architecture offers potential for future expansion into HR management functionalities.

2. OBJECTIVE

To develop a secure and efficient contact management system that enables users to store, organize, and access contact information seamlessly. The application will provide features for adding, editing, and searching for contacts, as well as sending emails directly from the platform. By prioritizing user experience and data security, the system aims to streamline communication and enhance productivity for individuals and small teams.

3. KEY FEATURES

Our application is going to be designed and developed with features like:

- * **Secure Login:** Users can access their contacts with ease through integrated Google and GitHub authentication.
- * **Contact Management:** Efficiently store and organize contact details, including names, addresses, phone numbers, and email addresses.
- * **Email Integration:** Easily send emails directly from the platform to contacts, saving time and effort.
- * **Reports Generation:** Easily generates the reports using Excel/PDF tools.

While primarily designed as a contact management tool, the system's foundation of data organization and communication capabilities provides a solid base for potential expansion into HR management functionalities. By incorporating modules for employee records, performance tracking, and recruitment, this platform can evolve into a comprehensive HR solution tailored to meet the specific needs of organizations.

4. DESIGN:

4.1: Flow Execution of SWIFTCONNECT:

The flow execution of the SwiftConnect includes User Creation/ Login to interacting with UI to access the features of application along with view of Profile and other operations like add, edit, delete contacts or sending emails or exporting the contacts etc.

Steps involved are as below:

*** User Access:**

- * User visits the application's homepage.
- * User chooses to sign up or log in.

*** User Authentication:**

*** Sign Up:**

- * User enters email, password, and other required information.
- * Application validates input and creates a new account.

*** Log In:**

- * User enters email and password.
- * Application verifies credentials and grants access.

*** Social Login:**

- * User clicks on a social login button (e.g., Google, GitHub).
- * Application redirects to the social network's authentication process.
- * Upon successful authentication, the user is granted access.

*** User Dashboard:**

- * User is redirected to their dashboard.
- * Dashboard displays an overview of contacts, groups, and other relevant information.

*** Contact Management:**

*** View Contacts:**

- * User can view a list of contacts.

*** Add Contacts:**

* User fills out a form with contact details (name, email, phone, etc.) and optionally uploads a picture.

- * Application saves the new contact.

*** Edit Contacts:**

- * User can modify existing contact details.

*** Delete Contacts:**

- * User can remove contacts.

*** Email Functionality:**

*** Compose Email:**

- * User can create a new email.
- * User can add recipients, subject, body, and attachments.

*** Send Email:**

- * Application sends the email using an email service provider (e.g., Gmail, SMTP).

*** Export Contacts:**

*** Export to Excel/ PDF:**

- * User can export their contact list to an Excel file or PDF file.

Additional Flows:

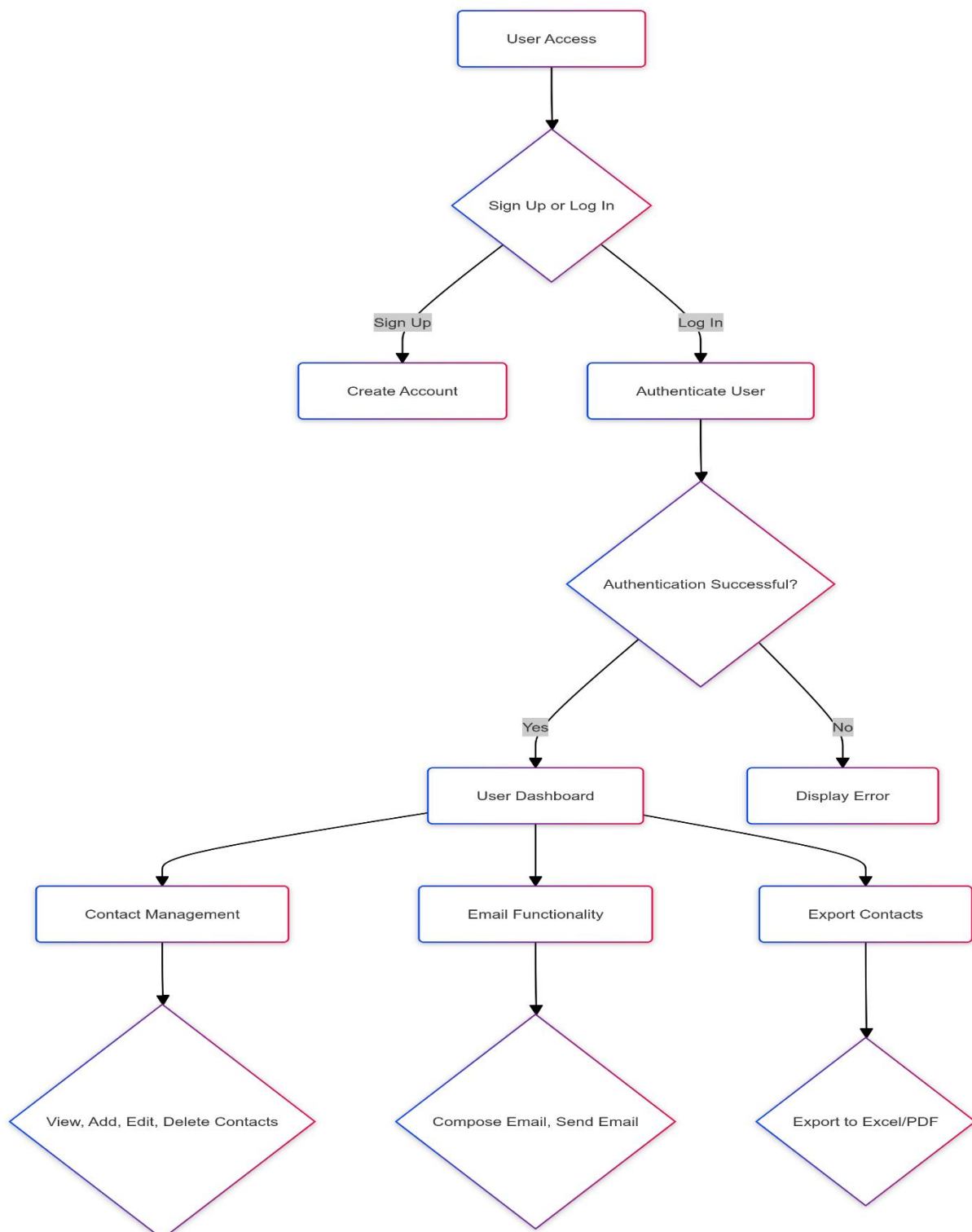
*** Profile Management:**

* User can view and update their profile information.

*** Search Functionality:**

* User can search for contacts.

The below FlowChart shows the key functionalities and flow of the application.



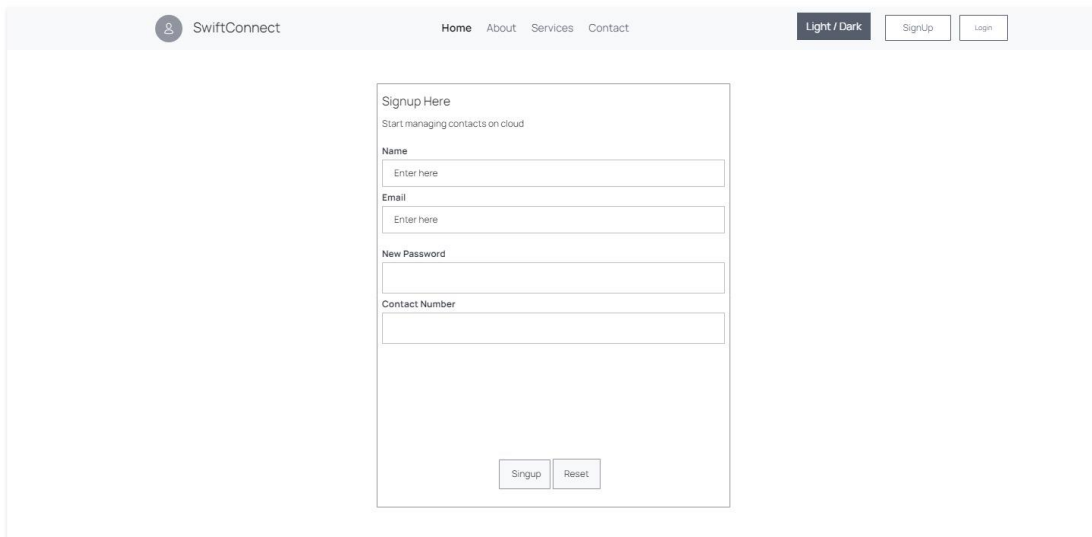
Flowchart for Swift Connect

4.2: UI wireframes or pictorial representation:

The UI will be designed using HTML, CSS (Tailwind). The below wireframes, represents the designed structure of the UI of the application, this design will be used for development of the application UI.

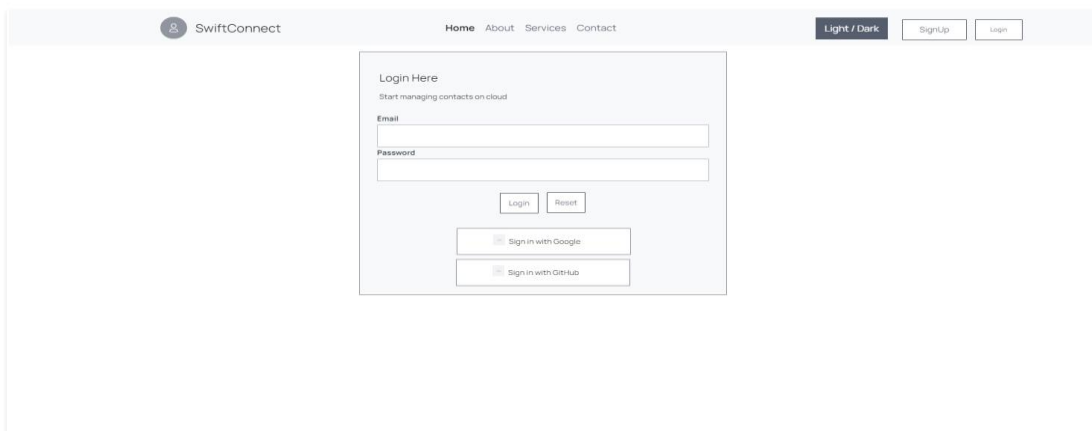
Below are few pages, covering most of the UI part of the application.

4.2.1: SignUp Page:



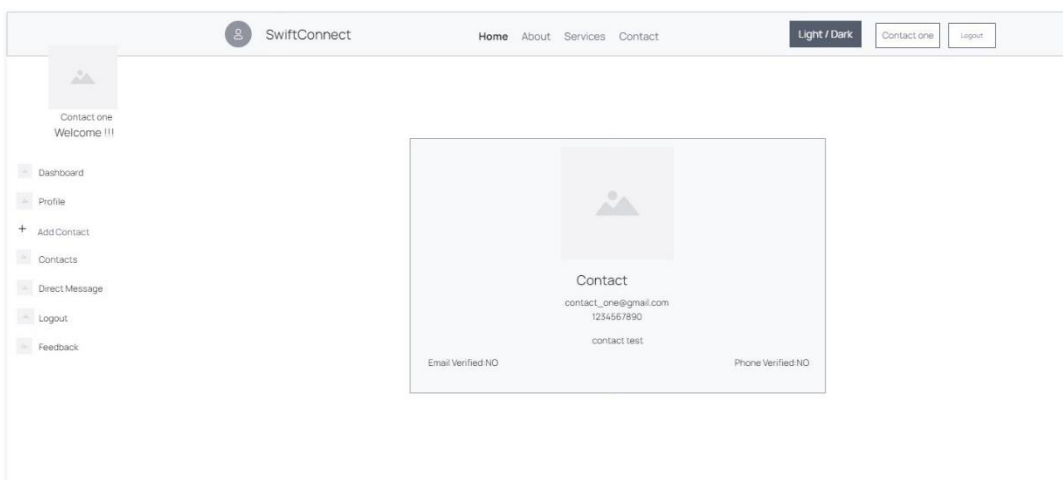
The wireframe for the SignUp page features a header with the SwiftConnect logo, navigation links (Home, About, Services, Contact), a Light/Dark theme toggle, and links to Sign Up and Log In. The main content area contains a 'Signup Here' section with the subtext 'Start managing contacts on cloud'. It includes input fields for Name, Email, New Password, and Contact Number, followed by 'Singup' and 'Reset' buttons.

4.2.2: Login Page:



The wireframe for the Login page has the same header as the SignUp page. The main content area contains a 'Login Here' section with the subtext 'Start managing contacts on cloud'. It includes input fields for Email and Password, followed by 'Login' and 'Reset' buttons. Below these are buttons for 'Sign in with Google' and 'Sign in with Github'.

4.2.3: Profile Page:



The wireframe for the Profile page features a header with the SwiftConnect logo, navigation links (Home, About, Services, Contact), a Light/Dark theme toggle, and links to Contact one and Logout. On the left is a sidebar with a user profile icon, a 'Contact one Welcome !!!' message, and a list of menu items: Dashboard, Profile, Add Contact, Contacts, Direct Message, Logout, and Feedback. The main content area displays a 'Contact' card with a profile picture placeholder, the name 'Contact', email 'contact_one@gmail.com', phone number '1234567890', and a 'contact test' message. At the bottom of the card are status indicators: 'Email Verified NO' and 'Phone Verified NO'.

4.2.4: Add new contact Page:

SwiftConnect

HomeAboutServicesContact

Light / DarkContact oneLogout

Dashboard

Profile

Manage

Add Contact

Contacts

Direct Message

Logout

Feedback

Add New Contact

This contact will be stored on cloud, you can direct email this person.

Contact Name

Contact Email

Contact Phone

Contact Address

Address of the contact

Contact Description

Write about your contact

Contact Image

Choose File

is this contact is your best friend?

Add Contact

Reset

4.2.5: Search & Export Page:

SwiftConnect

HomeAboutServicesContact

Light / DarkContact oneLogout

Dashboard

Profile

Manage

Add Contact

Contacts

Direct Message

Logout

Feedback

All Your Contacts

List of all contacts.

Search for contacts

Search

Export

NAME	PHONE	LINKS	ACTION
<div>Test</div> <div>test@gmail.com</div>	1234567890		<div>Edit</div> <div>Delete</div> <div>View</div>
1			

4.3 Backend:

The application backend part will be done using Spring Framework and its features. It will be implemented with the above mentioned features of the application.

5. TECHNOLOGY STACK

Spring Boot: Provides a foundation for building the application, offering auto-configuration, dependency injection, and embedding a servlet container.

Spring Framework: Underpins Spring Boot, offering comprehensive support for dependency injection, aspect-oriented programming, and data access.

Spring MVC: Handles web requests and responses, providing a flexible and efficient model-view-controller architecture.

Spring Data JPA: Simplifies data access by providing a repository abstraction layer, reducing boilerplate code and promoting database independence.

Validation: Ensures data integrity by validating user input, preventing errors and improving application reliability.

Spring Security: Protects application resources by implementing authentication and authorization mechanisms.

MySQL/Postgresql: Stores application data, providing reliable and efficient data management.

Java Email Services: Enables email communication within the application for notifications and other purposes.

PDF/Excel Tools: Generate reports in various formats for data analysis and presentation.

Other technologies like **Thymeleaf** - which Renders dynamic HTML templates, offering natural template syntax and integration with Spring MVC ,

Javascript, TailwindCSS - to Create interactive and visually appealing user interfaces are also going to be used in the application for enhancing user experience.

6. DEVELOPMENT METHODOLOGY

Requirements Gathering: Define system requirements, user stories, and functional specifications.

Database Design: Create a well-structured database schema, considering data normalization and indexing for optimal performance.

Entity Modeling: Develop domain models representing data entities, leveraging Spring Data JPA annotations for mapping to database tables.

Repository Implementation: Create repository interfaces using Spring Data JPA to interact with the database, minimizing data access logic.

Service Layer: Define business logic and application services, encapsulating core functionalities.

Controller Development: Build RESTful controllers using Spring MVC to handle incoming requests and generate appropriate responses.

UI Development: Design and implement user interfaces using Thymeleaf, JavaScript, TailwindCSS, for an intuitive and visually appealing experience.

Security Implementation: Integrate Spring Security to protect sensitive data and restrict unauthorized access.

Email Integration: Configure email services to enable sending and receiving emails within the application.

Report Generation: Implement PDF/Excel generation capabilities to create informative reports.

Testing: Conduct thorough unit, integration, and system testing to ensure quality and reliability.

Deployment: Deploy the application on suitable platform.

Special Features:

Database Independence: Spring Data JPA allows for easy switching between different databases without significant code changes.

Rapid Development: Spring Boot's auto-configuration and convention over configuration approach accelerates development.

Security: Spring Security provides robust protection against common vulnerabilities.

Scalability: Cloud-based file storage (AWS/Cloudinary) offers flexibility and scalability for handling increasing data volumes.

User Experience: Modern UI frameworks like TailwindCSS enhance the user interface.

Data Export: PDF and Excel report generation provides valuable data analysis and presentation capabilities.

By following this methodology and leveraging the strengths of the chosen technologies, we aim to build a high-quality, scalable, and user-friendly contact management system.

7. BENEFITS:

By leveraging the power of Spring Boot and other technologies, this application aims to deliver a reliable, efficient, and user-centric solution for managing contacts.

- * **User-Friendly Interface:** Intuitive design for easy navigation and use.

- * **Scalability:** The application can be adapted to accommodate growing contact lists and team sizes.

- * **Customizability:** Users can personalize their contact views and preferences.

- * **Data Security:** Robust security measures protect sensitive information from unauthorized access.

- * **Integration Capabilities:** Potential for integration with other business applications (e.g., HR systems, email clients).

- * **Secure Data Storage:** Protect sensitive contact information through secure login and data encryption

8. LIMITATIONS & SOLUTIONS:

Limitations and potential solutions for our application are:

- * Using Spring Boot and managing the project may require some learning and familiarity with the framework.

- * Security: As of now we are using spring security and authorization, to prevent application from security risks. But will try to provide more enhancements to make the application enabled for large scale and to keep it away from potential risks. For now, measures like data encryption and secured login are already considered.

- * Scalability: As the number of contacts grows, the system might face performance issues if not designed to handle large datasets efficiently.

For this we are limited to small scale for now, but performance optimizations can be done with proper DB usage like using indexing in databases, also will try to get our application on cloud for better Scalability.

- * Customizable Reports: The ability to generate reports using Excel/PDF tools is there, but there might be limitations in customizing these reports to meet specific needs. So, we can try enhancements later to allow users to customize report templates and choose specific data fields to include, making reports more relevant to their needs.

For now,

Scope is limited to small scale, but there will always be room for improvement and moving the application to large scale , along with data Privacy enhancements can be done as some enhancements.

Which is not considered in scope as of now.

We are going to use open source tools and technologies like Java, Spring Boot, Spring Security, MySQL etc., which provides cost effective and Platform independent solution.

9. DETAILED PLAN OF WORK

The following table summarizes the detail plan of work against the planned duration in weeks

Serial No.	Tasks or subtasks to be done (be precise and specific)	Planned duration in weeks	Specific Deliverable in terms of the project	Status
1	Requirement Gathering and Analysis	1 - 2	Requirement Analysis and brainstorming sessions to figure out potential issues or limitations and work upon the ideas to fix them.	Completed
2	Design	3 - 5	-Designing UI Pages and learning basic working of tools and technologies to be used for application development. -Worked on UI designs and deciding the flow of application.	Completed
3	Coding & Integrations	6 – 12	Building Front end and Back- end of application. -Started working on the Home page of the application.	In Progress
4	Unit Testing and integration Testing	13 – 15	Preparation of test cases and testing the application	Yet to Start
5	Deployment & Documentation	16	Final dissertation report.	Yet to Start

10. REFERENCES:

- * **Spring Boot:** <https://spring.io/projects/spring-boot/> , [Spring Boot Tutorial - Bootstrap a Simple App | Baeldung](#)
- * **Spring Data JPA:** <https://spring.io/projects/spring-data-jpa/> , , [Introduction to Spring Data JPA | Baeldung](#)
- * **Thymeleaf:** <https://www.thymeleaf.org/> , [Tutorial: Thymeleaf + Spring](#)
- * **Spring Security:** <https://spring.io/projects/spring-security/> , [Spring Security Tutorial - GeeksforGeeks](#)
- * **JavaMail API:** <https://javaee.github.io/javamail/docs/api/> , [Java Email Sender: Sending Emails Smoothly with Spring Boot | by José Robson | Medium](#)
- * **Hibernate Validator:** <https://hibernate.org/validator/> , [Hibernate Validator Specific Constraints | Baeldung](#)
- * **MySQL:** <https://www.mysql.com/> , [Integrating MySQL with Spring Boot: A Comprehensive Guide: All in ONE \[P-I\] | by Rabinarayan Patra | Medium](#)
- * **PostgreSQL:** <https://www.postgresql.org/> , [Spring Boot with PostgreSQL: A Step-by-Step Guide | by Pabitra Priyadarshini Jena | Medium](#)
- * **JavaScript:** <https://developer.mozilla.org/en-US/docs/Web/JavaScript/> , [Client Side Development with Spring Boot Applications](#)
- * **Tailwind CSS:** <https://tailwindcss.com/> , [Adding Tailwind CSS to the Spring Boot Application | by Javad Yadegari | Medium](#)
- * **Apache POI (for generating Excel files):** <https://poi.apache.org/> , [Working with Microsoft Excel in Java | Baeldung](#)
- * **iText (for generating PDF files):** <https://itextpdf.com> , [Creating a PDF Export API with iText PDF in Java Spring Boot | by Rijul Dahiya | Medium](#)