InstaVibe

By:

Iqra Asif (066314)

Muqaddas Shahnawaz (066293)

Malaika Ali (066307)

Project Supervisor:

Mrs Lubna Zakia

Bachelor of Science in [Information Technology] (2021-2025)

FACULTY OF COMPUTING &

INFORMATION TECHNOLOGY (FCIT),

UNIVERSITY OF THE PUNJAB, LAHORE.

A project presented to

University of the Punjab, Lahore

In partial fulfilment

of the requirement for the degree of

Bachelors of Science in [Information Technology] (2021-2025)

By:

Iqra Asif (066314)

Muqaddas Shahnawaz (066293)

Malaika Ali (066307)

FACULTY OF COMPUTING &

INFORMATION TECHNOLOGY (FCIT),

UNIVERSITY OF THE PUNJAB, LAHORE.

DECLARATION

We**, Iqra Asif**, Roll No. 066314, **Muqaddas Shahnawaz**, Roll No. 066293 and **Malaika Ali**, Roll No. 066307, Session (2021-2025) from Department of Information Technology, F.G Post Graduate College for Women Kashmir Road, Rawalpindi, hereby sincerely declare that the work submitted in this research report entitled "**InstaVibe"** is our own work. This work has been completed at the Department of Information Technology, F.G Post Graduate College (Kashmir Road), Rawalpindi under the supervision of Ma'am Lubna Zakia. It has not been previously presented to any other institution or university for the degree.

Signature: ----------

Iqra Asif [066314]

Signature: ----------

Muqaddas Shahnawaz [066293]

Signature: ----------

Malaika Ali [066307]

Certificate for Approval

It's certified that final year design project (FYDP) of "**InstaVibe"** was developed by **Iqra Asif** (Roll No. 066314), **Malaika Ali** (Roll No. 066307), and **Muqaddas Shahnawaz** (Roll No. 066293) under my supervision of **"Mrs. Lubna Zakia"** in my opinion; it is fully satisfactory in scope and quality for the degree of Bachelors of Science in [Information Technology]

Signature: ----------

FYDP Supervisor: Mrs. Lubna Zakia

Signatures (Faculty Advisory Committee (FAC)

Signatures			
Name			
	FAC1	FAC2	FAC3

Signature: ----------

Head of FYDP Coordination Office:

Signature: ---------- Dated: _____

Chairperson, Department of [Information Technology]

Acknowledgement

First of all, we thank Allah Almighty for giving us the strength, knowledge, and patience to complete this project. Without His blessings, this would not have been possible.

We are also very thankful to our respected supervisor, **Mrs. Lubna Zakia, Sir Sheheryaar Afzal** for his kind guidance, helpful suggestions, and continuous support during this project. His help made a big difference in our work.

Lastly, we are very grateful to our parents and family. Their love, prayers, and support gave us the energy and motivation to keep going.

Signature: ----------

Iqra Asif [066314]

Signature : ----------

Muqaddas Shahnawaz [066293]

Signature: ----------

Malaika Ali [066307]

Project in Brief

Project Title	**InstaVibe**
Objective	The objective of this project is to create a secure and scalable social media platform that enables users to share multimedia content, interact, and receive personalized recommendations. The platform will prioritize efficient data management, and perfect user engagement. Additionally, it aims to ensure high performance, and responsiveness for a smooth and dynamic user experience
Undertaken By	• **Iqra Asif,** Roll No. (066314) • **Muqaddas Shahnawaz,** Roll No. (066293) • **Malaika Ali,** Roll No. (066307)
Audited By	Ma'am Lubna Zakia
Date Started	22-10-2024
Date Completion	continue
Language and Technology Used	• HTML, CSS, JavaScript • C# (.net framework) • MS SQL Server 2019
Tools Used	Visual Studio 2022,SQL Server Management studio
Operating System	Microsoft Windows 10

Table 1.1. Project Brief Description

Table of Content

Chapter 1:
Introduction

1.Introduction:

The purpose of the **InstaVibe** is to provide users with an enjoyable environment for sharing, interacting, and connecting. Inspired by Instagram, it allows for smooth media sharing, user interaction, and customized content discovery. It is possible to build profiles and add images and videos. Individuals can be followed, and users can message, like, and comment on each other's posts.

The goal of an **InstaVibe** is to be a dynamic and interesting social networking site. In today's digital environment, social media is essential for self-expression and communication. The main objective of this project is to enhance the user experience through secure authentication, real-time updates, and a user interface that is user-friendly.

1.1 Problem Statement:

In today's digital age, people want an easy and engaging way to share their moments, connect with others, and express themselves online. Popular social media platforms are often too complex, overloaded with features, or don't meet the specific needs of users. Our project, **InstaVibe**, aims to solve this by creating a user-friendly, fast, and secure platform where users can share photos and videos, interact through likes, comments, and messages, and manage their social experience with simple settings.

1.2Problem Solution:

To solve these problems, **InstaVibe** is introduced as a fresh and user-friendly social media platform. It offers a simple design that is easy to understand, even for beginners. Users can share posts quickly, interact smoothly, and discover content that matters to them. **InstaVibe** includes fun, interactive features that keep users engaged and happy. It focuses on providing a safe, and smooth experience.

1.3 Objectives of the Proposed System

Objectives of the Proposed **InstaVibe** system are as follows:

- To create a simple and easy-to-use platform where users can share photos, videos, and stories without confusion.
- To build a secure sign-up and login system that keeps user data safe and only allows verified users to access their accounts.
- To add real-time chat so users can send messages, pictures, and documents quickly, with most messages sent in less than 1 second.
- To provide features like likes, comments, and shares so users can interact with each other's posts and stay socially connected.
- To give users full control over their privacy by allowing them to choose who can see their posts and manage their account settings.

1.4. Scope of the project:

The scope of this **Instavibe** includes several key areas:

- **User Engagement: InstaVibe** will let users communicate, share content, and interact with likes, comments, and shares, creating an active online community.
- **Content Sharing:** Users can post and share photos, videos, and stories, making it a fun space for creativity and sharing experiences.
- **Real-Time Communication:** Group chats and direct messages will let users interact instantly, boosting social connections.
- **Media and Document Sharing:** Users can easily share photos, videos, and documents, making communication more dynamic.
- **Security and Privacy: InstaVibe** will keep user data safe with modern security feature like Login Activity Monitoring.

1.5 System Components

1.5.1 Module 1: Client Web App (InstaVibe Web App)

Features:

User Registration and Login: Users can sign up, log in, and manage their accounts.

Profile Management: Users can view and edit their profiles, including profile pictures and personal details.

Content Sharing: Users can upload and share photos, videos, and stories.

Engagement Features: Likes, comments, and sharing options for posts and media.

Real-Time Messaging: Group chats and direct messaging for instant communication. Media Sharing: Users can share photos, videos, and documents within messages or posts.

Notifications: Alerts for new messages, comments, likes, etc.

1.6 Related System Analysis/Literature Review

InstaVibe will allow users to share photos, videos, documents, and stories. It will also have real-time messaging, group chats, and full privacy control. Existing apps either have limited privacy, no document sharing, or fixed story time.

Table 1- 1 Related System Analysis with proposed project solution

Application Name	Weakness	Proposed Project Solution
InstaVibe	Limited privacy, no document sharing, fixed story duration.	InstaVibe will offer full privacy settings, allow document sharing, and story time control.

1.7 Vision Statement

Our vision is to create a dynamic, secure, and easy-to-use social media platform where people from all walks of life can connect, share, and engage meaningfully. InstaVibe aims to become a trusted digital space that encourages creativity, self-expression, and genuine social interaction through features like photo/video sharing, real-time messaging, and personalized experiences. We envision InstaVibe as a future-ready platform that not only keeps up with modern technology but also promotes a safe, inclusive, and user-driven community. By continuously improving our platform, we aspire to offer users a smooth, enjoyable, and empowering social networking experience that they can rely on every day.

1.8 System Limitations and Constraints

Limitations:

- Slow internet can affect image and video uploading.
- App may not work the same on all devices.
- We can't fully control what users post.

Constraints

- Only main features will be developed first (like post sharing, chatting, and privacy settings).
- Only the web version is being developed now; the mobile app will be made later.
- Due to limited time, budget, and a small team, not everything will be built at once
- Cool features like filters or AI are not planned for now.

1.9 Tools and Technologies

Tools and Technologies	**Tools**	**Version**	**Rationale**
	1.Visual Studio 2.SQL Server	2022 2019	To write, test, and manage the code for the web app. To store user data like profile, post and messages.
	Technology	**Version**	**Rationale**
	Asp .Net MVC with Razor pages for frontend. C#	4.0 5.0	To build the structure of the website using the MVC pattern. Used for writing the backend logic with features like async and await for better performance.

1.10 Project Deliverables

Project Plan: A document showing what we will do and when.

Requirements Document (SRS): A document explaining what the app will do.

System Design: A plan for how the app will work and its features.

Prototypes: Simple designs of how the app will look.

Working App: The InstaVibe app with basic features like posting and chatting.

Project Report: A report explaining the whole project.

User Guide: A guide to help users understand how to use the app**.**

1.11 Summary

In this chapter, we discussed the basic details of the InstaVibe project, like its goals, features, scope, planning, and tools. This chapter is important because it sets a clear direction to successfully complete the project and achieve its main objectives.

Chapter 2:

Requirements Analysis

1. Analysis

In this chapter, we will carefully study the needs of the **InstaVibe** project. We will understand what the users want, how the system should work, and what features are important. This analysis will help in planning and building the project properly.

1.1 User classes and characteristics

User class	Characteristics
Registered users	-Create an account and login to use the app -Upload image and videos to share with others -Like and comment on posts. -Follow and unfollow other users. -Send and receive private messages. -Customize profile setting (e.g theme change, password update) -Use the search feature to discover user or posts.
Non-Registered users	-Can visit the platform but have very limited access. -Cannot like ,upload, send messages. -Can view some public content. -need to register and login to use full features.

2.2. Requirement Identifying Technique

2.2.2 Use-case Model:

A **Use Case Model** shows what the new system will do. Each **Use Case** explains one complete task that a user (a person) can do using the system. It describes how the user interacts with the system to get something useful done — like signing in, uploading a photo, or editing a profile.

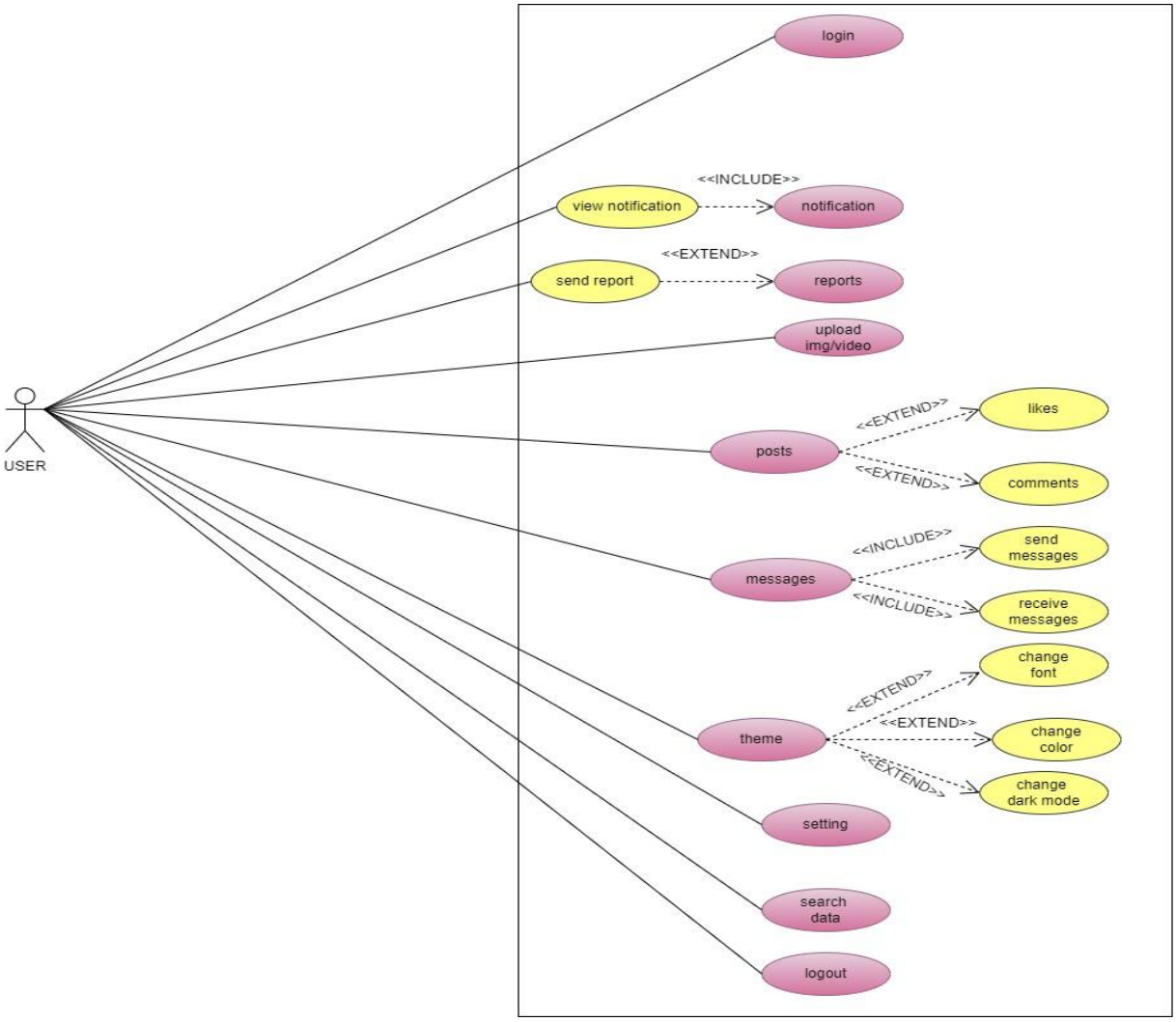


Fig 1.1. Use Case Diagram

2.2.3. Use Case description:

2.2.3.1. Login

Use Case ID:	**UC-01**
Use Case Name:	Login
Description:	The user enters their username or email and password to log in. The system checks the information, and if correct, allows the user to enter their account and shows the home page.
Actor	Registered User.
Pre-condition:	User must enter correct username/email and password, and account should be active.
Post-condition:	If correct, user logs in; if not, error is shown.
Includes:	Visual Studio 2022, SQL Server 2019 Database.
Basic Flow:	• User enters username/email and password. • System checks and verifies them. • If correct, user logs in and goes to home page. • If wrong, error message appears.

Frequency of use:	Multiple times daily, depending on users' tasks.
Exception	If login fails, show "Invalid user data."

Table 1.2. Login Use Case Description

2.2.3.2. Registration

Use Case ID:	**UC-02**
Use Case Name:	User Registration
Description:	A new user fills out the sign-up form with details like name, username, email, and password. The system checks the data, saves it to the database if valid, and shows a success message.
Actor:	New User
Pre-condition:	User must enter all required fields correctly (e.g., valid email, strong password). Username and email must not already exist.
Post-condition:	If successful, the account is created and ready for login. If not, the user is shown a clear error message.
Includes:	Visual Studio 2022, SQL Server 2019 Database.
Basic Flow:	• User opens the registration page • Fills out the form with name, username, email, and password • System checks the input (client-side + server-side) • If everything is valid and unique, system saves data in database • Success message appears; user may now log in
Frequency of use:	Once per new user (or rarely, if someone re-registers after deleting an account)
Exception:	If username/email already exists, show: "Username or email already exists." If required fields are empty or invalid, show: "Please fill out all fields correctly."

Table 1.2. Registration Use Case Description

2.2.3.3. Notification

Use Case ID:	**UC-03**
Use Case Name:	Notification
Description:	Users can see, mark as read, and delete notifications. Viewing and sending are part of notifications, while marking, updating, and deleting are extra actions.
Pre-condition:	Users must be logged in and have notifications to view, mark as read, or delete."
Post-condition:	The notification status is set to 'read' when viewed or removed when deleted.
Includes:	Visual Studio 2022, SQL Server 2019 Database.
Basic Flow:	• User logs in to the system. • User receives a notification in their inbox.

	• User marks the notification as read or deletes it if no longer needed.
Frequency of use:	Users frequently receive and manage notifications daily.
Alternative Flow:	For Users, if no new notifications are available, the inbox does not change.
Exception:	Exceptions include network issues preventing retrieval or failure to mark as read.

Table 1.3. Notification Use Case Description

2.2.3.4. Report

Use Case ID:	**UC-04**
Use Case Name:	Report
Description:	Users can create, and send reports, like reporting a post.
Pre-condition:	Users must be logged in to generate, and send a report.
Post-condition:	The report is submitted after the action.
Includes:	Visual Studio 2022, SQL Server 2019 Database
Basic Flow:	• User logs in to the system. • User generates and submits a report. • System confirms report submission and notify them.
Frequency of use:	Users use reports sometimes, mostly when they want to report a problem.
Alternative Flow:	If report submission fails due to a network issue, users can retry later.
Exception:	Users may face report submission failures from network issues.

Table 1.4. Report Use Case Description

2.2.3.5. Upload image and video

Use Case ID:	**UC-05**
Use Case Name:	Upload image and video
Created By	Iqra Asif, Malaika Ali &Muqaddas Shahnawaz.
Description:	Users can pick and upload pictures or videos. The system saves them, makes them look better.
Pre-condition:	The user need to be logged into the system with permission to upload media. The platform must support the allowed file types, and enough storage space must be available.
Post-condition:	After a successful upload, the image or video is stored in the database and made available for viewing or sharing. A confirmation message is displayed to the user.
Includes:	Visual Studio 2022, SQL Server 2019 Database.
Basic Flow:	• The user logs into the system. • The user goes to the Post section. • The user selects an image or video file from their device. • The system checks the file format and size. • The file is uploaded to the database. • A confirmation message is displayed, and the uploaded file becomes available for viewing or sharing.

Frequency of use:	Users often upload images and videos, based on how active they are. Most uploads happen during busy times, like events or trending topics.
Alternative Flow:	If the upload fails, it try to upload again.

Table 1.5. Upload image and video Use Case Description

2.2.3.6. Like and comment on posts

Use Case ID:	**UC-06**
Use Case Name:	Like and comment on posts.
Created By	Iqra Asif, Malaika Ali &Muqaddas Shahnawaz.
Description:	The Like and Comment feature lets users interact with posts by liking or commenting. The system tracks these actions and notifies the post owner when someone likes or comments.
Pre-condition:	The user must be logged into the system and have access to view the post. The post must exist and be available for interaction (i.e., not deleted or restricted).
Post-condition:	After a like or comment, the system saves the action, updates the post, and informs the owner. The post shows the new like or comment .
Basic Flow:	• The user logs into their account. • The user finds a post they want to like or comment on. • The user clicks the Like button or types a comment. • The system checks and saves the like or comment. • The post owner gets a notification (if it's turned on).
Frequency of use:	Users like and comment on posts frequently, especially on popular content. More people interact with posts when something popular is happening.
Alternative Flow:	If a like fails due to a network issue, the user can retry later. If a comment contains restricted words, the system asks for edits before posting.
Exception:	If the internet disconnects, the like or comment doesn't go through. If a comment contains banned words, the system blocks it.

Table 1.6. Like and comment on posts Use Case Descriptio

2.2.3.7. Send and receive messages

Use Case ID:	**UC-07**
Use Case Name:	Send and Receive Message
Created By	Iqra Asif, Malaika Ali &Muqaddas Shahnawaz
Description:	The Send and Receive Messages use case allows users to communicate through direct or group messaging. The system sends messages quickly, saves them safely, and alerts users when a new message arrives.
Pre-condition:	The user has to be logged in, and the person they're messaging should have the messaging feature turned on.

Post-condition:	The message is saved in the system, sent to the recipient. The recipient is notified as soon as the message is delivered.
Basic Flow:	• The user logs into the system. • The user goes to the messaging section and picks a recipient. • The user types a message and clicks Send. The system saves the message in the database. • The message is delivered to the recipient either instantly or when they are online. The recipient gets a notification (if enabled).
Frequency of use:	Users send and receive messages frequently, depending on activity and chats. Message exchange is higher during peak times, like evenings and weekends.
Alternative Flow:	If a message fails to send due to no internet, the system retries when connected. If the recipient is offline, the message is delivered when they come online.
Exception:	If the internet is disconnected, the message fails to send. If the recipient has blocked the sender, the message is not delivered.

Table 1.7. Send and receive message

2.2.3.8. Customize theme

Use Case ID:	**UC-08**
Use Case Name:	Customize theme
Created by	Iqra Asif, Malaika Ali &Muqaddas Shahnawaz
Description:	The Customize Theme feature lets users adjust the font size, color, and layout what they like.
Pre-condition:	To customize the theme, the user needs to be logged in and have access to theme settings. The system should allow changes to font size, color, and mode.
Post-condition:	After customizing the theme, the new font size, color, or mode settings are saved and updated in the user's interface.
Frequency of use:	The Customize Theme feature is used sometimes, as users usually set their choices once and change them when needed.
Alternative Flow:	If a user picks a font, color, or mode that isn't supported, the system shows a default option. If the changes don't work, the user can try again or go back to the default settings
Exception:	If the theme customization fails due to a system error the user receives an error message and the previous theme remains unchanged.

Table 1.8. Customize Theme

2.2.3.9. Search data

Use Case ID:	**UC-09**
Use Case Name:	Search data

Created By	Iqra Asif, Malaika Ali &Muqaddas Shahnawaz
Description:	The Search Data feature helps users find posts, people, or messages by typing keywords. The system searches and shows the best results.
Pre-condition:	The user should be logged in and able to use the search option. Also, the data they are looking for must be available in the system.
Post-condition:	The system checks the search, finds the matching results, and shows them to the user. It may also save the search history if needed.
Basic Flow:	• The user opens the app and goes to the search bar. • The user types a word to search. • The system checks the database and finds matching results. • The system shows the results to the user. • The user can look at, change, or interact with the results.
Frequency of use:	Users often search to look for posts, friends, or chats.
Alternative Flow:	If no exact match is found, the system suggests similar results. If the search fails due to a network issue, the user can retry later.
Exception:	If there's a search issue due to a network problem, the user can try again later..

Table 1.9 Search data

2.3 Functional Requirements

Functional requirements describe the specific functions or capabilities that the software system should perform. This includes features like user authentication, data input validation, data manipulation and reporting. Functional requirements specify what the system should do. Here are some functional requirements for the **InstaVibe**:

1. Post:

- Users should be able to create, edit, delete, and share posts.

- Posts can include text, images, videos, and links.

- Posts should have settings to control who can see them, like public, private, or only specific groups.

2. User:

- Users should be able to create an account, log in, and manage their profile (e.g., username, bio, photo).

- Users can follow or unfollow other users, and view their feeds.

- Users should be able to send and receive messages with other users.
- Users can manage privacy settings (who can see their posts, messages, etc).

3. Message:

- Users should be able to send private messages or group messages.
- Messages should support text, images, and files.
- Notifications for new messages should be Shown.

4. Theme:

- Users should be able to customize the app's theme (e.g., dark mode or light mode).
-Users should be able to customize the app's theme like font size or color.
-Theme settings should be saved for each user.

5. Settings:

- Users should be able to manage their account settings (e.g., password, notifications).
- Privacy settings (who can see posts, send messages, etc.).
- Notification setting (email, instant messages etc.).

6. Search:

- Users should be able to search for other users, posts by typing keywords.
- Recent searches should be stored so users can quickly access them later.

7. Notification:

- The system should alert users when they receive new messages, likes, or comments.
- Users should be able to turn off or mute notifications for certain activities.

8. Comments:

- Users should be able to comment on posts and reply to other users comments.
- Users are notified when someone replies to their comment.
- Option to delete one's own comments on the posts.

9. Likes:

- Users should be able to like or unlike posts and comments.
- A notification or update should be sent when a user receives a like.
- Users should be able to see who liked their posts.

2.3.1 Functional Requirement

Table 1- 1 Description of FR-1

Identifier	FR-1
Title	Post creation
Requirement	The user should be able to create, edit, delete and share post including text ,images and videos.
Source	Users
Rationale	To allow user to interact with the app content such as posts, messages media upload and notification.
Business Rule	Must support text, image and video formats
Dependencies	FR-2
Priority	High

Table 1- 2 Description of FR-2

Identifier	FR-2
Title	User Profile
Requirement	The system should allow users to create an account, log in and update their profile (username, bio, photo)
Source	Users
Rationale	To give each user a space to manage their information and settings.
Business Rule	Username and email must be unique.
Dependencies	None
Priority	High

Table 1- 3 Description of FR-3

Identifier	FR-3
Title	Messaging
Requirement	The user should be able to send private/group messages with text, images and files
Source	Users
Rationale	Enable users to easily communicate and share information with others within the app.
Business Rule	The message should be sent and received instantly, without delay.
Dependencies	FR-2
Priority	High

Table 1- 4 Description of FR-4

Identifier	FR-4
Title	Notification
Requirement	The system will instantly notify users about likes, comments, and messages
Source	Users
Rationale	Keep user informed about new actions, like likes, comments, or messages, so they get the latest information.

Business Rule	Users should be able to adjust their notification settings to control what alerts they receive.
Dependencies	FR-1, FR-2, FR-3.
Priority	Medium

Table 1- 5 Description of FR-5

Identifier	FR-5
Title	Theme Customization
Requirement	The user should be able to switch between light/dark mode and adjust font size and color.
Source	Users
Rationale	Helps users feel more connected to the app by giving them control over its appearance.
Business Rule	Settings stay the same every time you log in
Dependencies	FR-2
Priority	Low

1.2 Non-Functional Requirements

Non-functional Requirements specifies the criteria that the system must meet in terms of performance, security, usability, reliability and other qualities. Non-functional requirements specify how the system should behave. Here are some non-functional requirements for the **Instavibe**:

1.Performance

Makes sure messages are delivered and the app responds quickly. It can handle many users at once without slowing down.

2. Security

Adds security features to protect messages and keep accounts safe.

3. Usability

Provides a simple, user-friendly interface for users to find things quickly and not get confused.

4. Privacy

The system protects user information and gives users the option to choose who can view their details, like profile picture and activity status.

5. Reliability

Make sure the system works well, even when alot of people are using it at the same time.

2.5 External Interface Requirements

The **InstaVibe** system will provide a responsive and user-friendly web interface that allows users to interact with various features such as login, registration, profile management, media

uploads, likes, comments, messaging, and post visibility options. The user interface will be developed using HTML, CSS, JavaScript, and integrated within the ASP.NET MVC framework. All interactions with the system will occur through web browsers on desktop. The application will communicate with a Microsoft SQL Server database using ADO.NET to handle data storage and retrieval for users, posts, messages, and notifications. The system will operate over standard HTTP/HTTPS protocols and be hosted on IIS (Internet Information Services) or a compatible web server. No specialized hardware is required to run the application, as it is designed for general-purpose computing devices with internet access.

2.5.1 User Interfaces Requirements

Design Standards: The app will follow material design for a modern and simple look.

Fonts, Icons, and Buttons: The main font used will be Roboto. Icons and buttons will be easy to understand and clearly labeled.

Colors: The app will have both light and dark modes, with easy-to-read colors**.**

Common Buttons: Buttons like Back, Home, and Settings will be available on all screens.

Message Display: Clear messages will appear for actions like success or errors, using icons and text.

2.5.2 Software interfaces

The app will connect to SQL Server 2019 for storing user data. It will use ASP.NET MVC 4.0 to control how the app works when users click buttons, open pages, or perform any task. The purpose of **ADO.NET** is to send and receive data between the app and the database.

2.5.3 Hardware interfaces

As a web app, the system will interact with standard hardware components like desktop computers, laptops. It will support keyboard and mouse inputs for interaction with the app. Users will need a working internet connection to send messages, upload posts, and refresh their feed. If any hardware like the Touchscreen, keyboard or internet is not available, the app will show an error message to inform the user.

2.5.4 Communications interfaces

Communication interfaces are the ways your app sends and receives data between the user's device and the server (or other systems). It's like a bridge that helps your app talk to the server. It uses HTTPS to safely send and receive data, like login details, posts, and messages. These interfaces help the app work smoothly in a web browser and allow real-time updates like new messages or likes without refreshing the page. They make sure users can use all features easily.

2.6 Summary

The requirements identification process involves gathering and defining the functional and non-functional needs of the system. This includes understanding user needs, system performance, and external interfaces. Use case analysis helps in mapping out how users will interact with the system, ensuring that all necessary actions and scenarios are covered. The requirements engineering process is crucial for the success of the project as it ensures all objectives are met and provides a clear direction for development.

Chapter 3:

Design and architecture

3. System Design:

The software will operate as a web application that interacts with web browsers (Chrome, Edge) and connects to a SQL Server database for storing user data. The system must also handle a high number of concurrent users without affecting performance.

3.1. Design consideration:

Assumptions and Dependencies:

The app relies on a consistent internet connection for user authentication. It uses SQL Server for storing user information, posts, and messages for handling multimedia files (images and videos).

Limitations:

The app may face performance issues when many users access it at the same time, leading to slower response times.

Risks:

Risks of data failures if sensitive information isn’t secured. It might not work well on all devices or browser.

3.2 Design Models

A design model in software development is a visual representation or abstraction of the system that outlines its structure, components, and interactions. It is used to communicate and understand how the system is built and how it will function once implemented.

1. Class Diagram:

A Class Diagram is a visual representation of class objects in a model system, categorized by class types. Each class type is represented as a rectangle with three compartments for the class name, attributes and operations. For **InstaVibe**, the class diagram provides a blueprint of the system's object-oriented design, helping developers understand how different components interact.

A Class Diagram of a **InstaVibe** represents the static structure of the system by showing the classes, their attributes, methods, and how they are related to one another. It is a key component of Object Oriented Design (OOD) and is used to illustrate the system's structure before moving to implementation. In the context of a **InstaVibe**, the class diagram serves to visualize how different components of the app interact and work together.

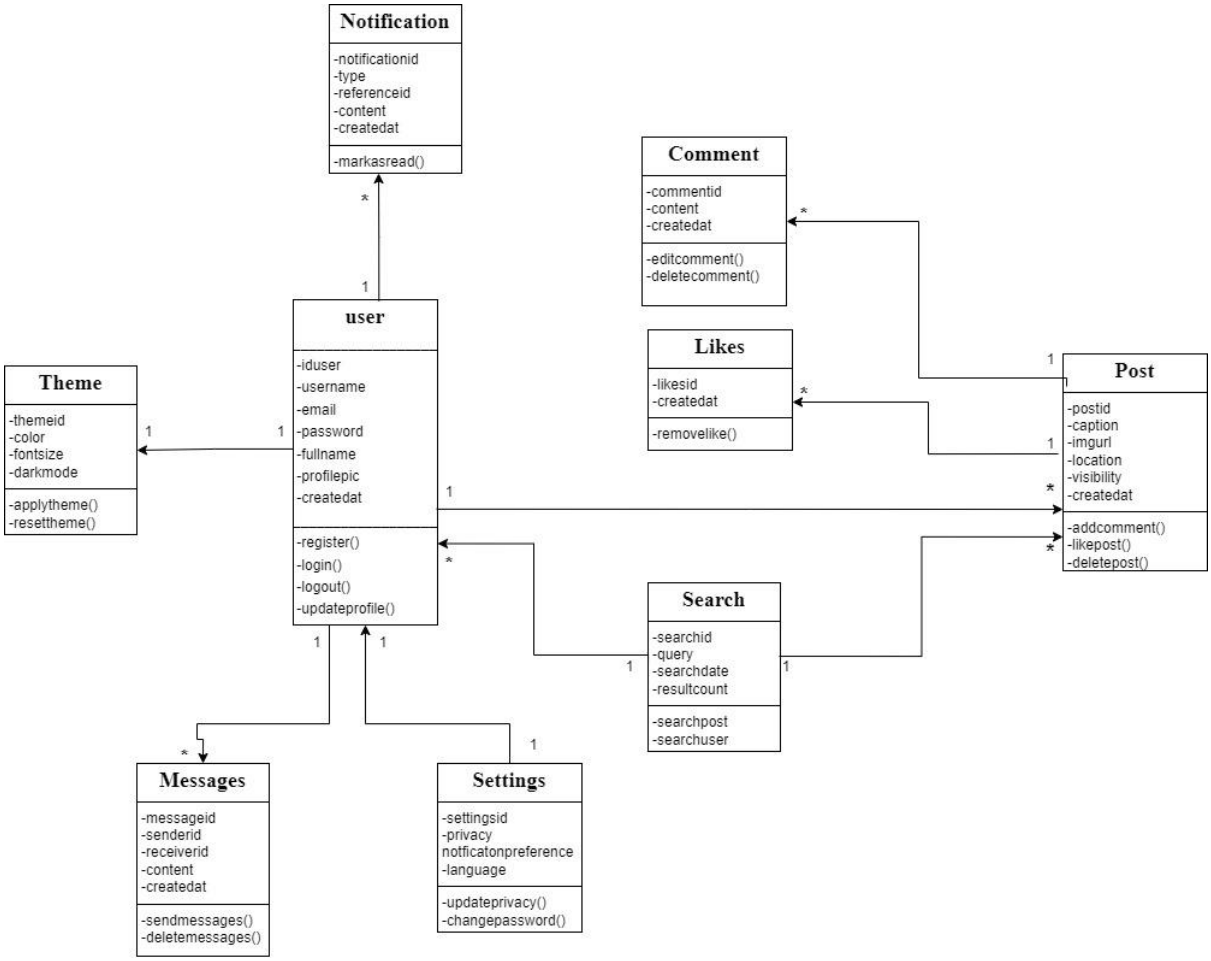


Fig 1.5. Class Diagram

2.Sequence Diagram

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

User-Side Sequence Diagram:

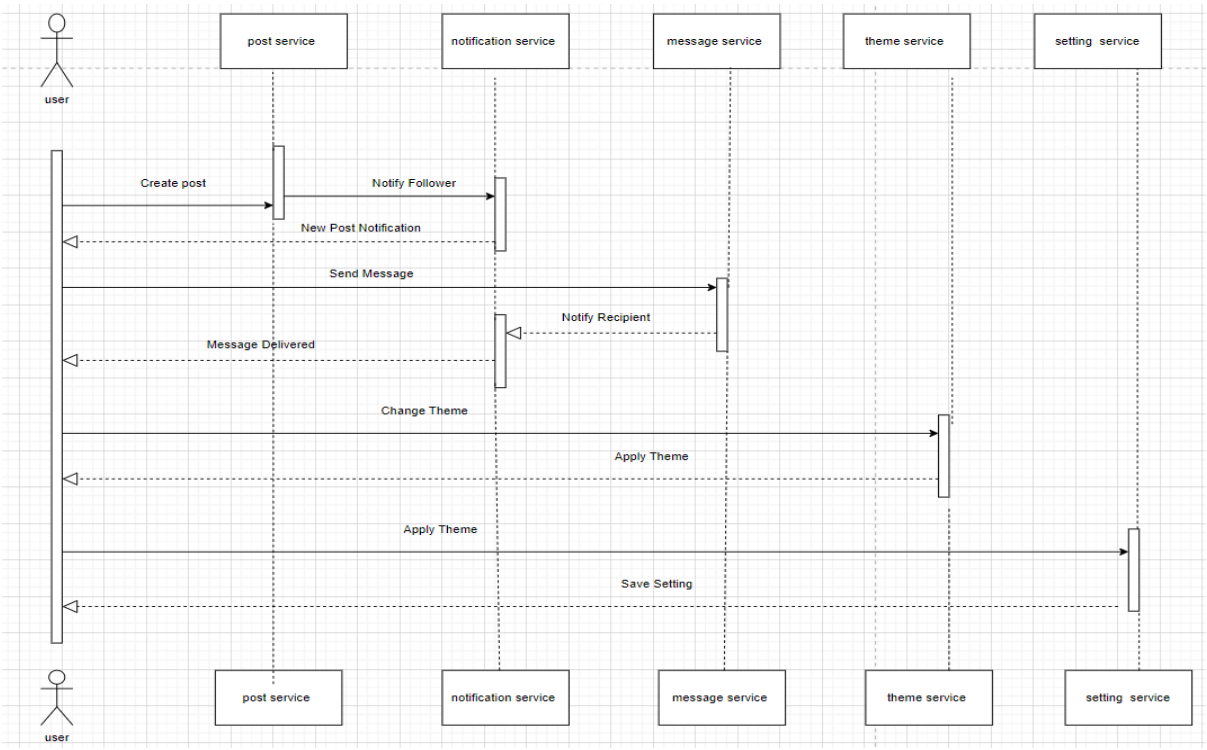


Fig 1.7 User-Side Sequence Diagram

3.1.7. State Transition Diagram:

A state transition diagram of a instavibe represents the different states a user and the system go through while interacting with the app. It typically starts with the login state, where the user either signs in or registers. Upon successful authentication, the app transitions to the home state, displaying recent chats and contacts.

When a user selects a contact, the app moves to the chat selection state, and once a conversation is initiated, it enters the active chat state, where messages can be sent and received. Messages further transition between states like sent, delivered, and read based on recipient actions. If the app supports calls, transitioning to a call state occurs when a voice or video call is initiated. The app may also enter a notification state when new messages arrive while the user is inactive. Finally, when the user logs out, the system transitions back to the logout state, completing the cycle.

3.1.7.1. User-Side State Transition Diagram:

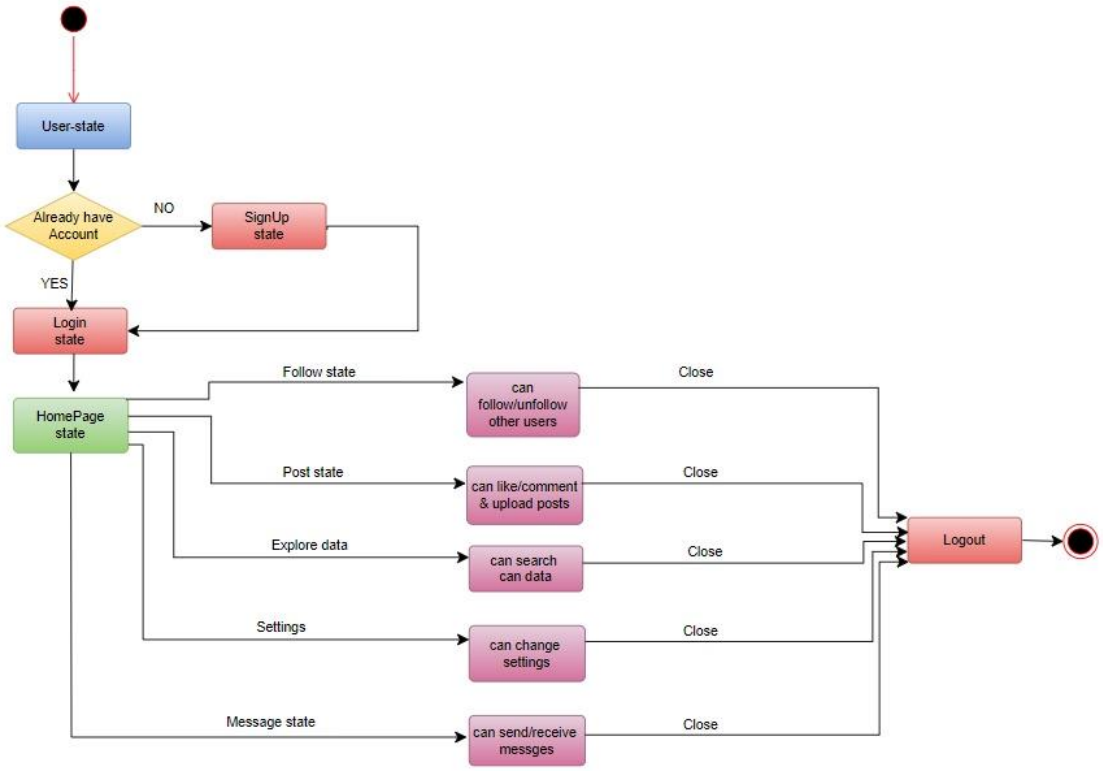


Fig 1.8 User-Side Sequence Diagram

3.3 Architectural Design

Architectural Design is the blueprint of the entire system. It shows how the software is divided into components, how they communicate, and what technology or design pattern (like MVC or Client-Server) is used.

InstaVibe follows the MVC (Model-View-Controller) architecture to keep the system well-organized and easy to manage.

- Model

Represents data and rules (like User, Message classes). Handles database-related logic using DatabaseHelper.cs.

_ View

The frontend seen by the user (HTML, Bootstrap, jQuery). Pages like SignUp, Login, Chat, and Profile are part of the View.

- Controller

Acts as a bridge between View and Model.Handles user actions (e.g., sending a message, updating profile). Examples: AccountController, ChatController, ProfileController.

Component Structure and Relationships

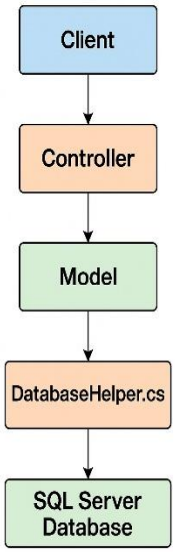
InstaVibe is a chatting web application based on Client-Server, MVC, Layered, and Multi-tiered architecture styles. The system is divided into 5 main components, each with a specific role:

Component	Role/Function
Client (View)	User screens (HTML, Bootstrap) for signup, login, chat, and profile.
Controller	Handles user actions and connects View with Model (Home, Account, Chat, Profile).
Model	Manages data (User, Message) and business rules.
Database Helper	Runs database queries and connects with SQL Server (using ADO.NET).
SQL Server Database	Stores users, chats, and profile data permanently.

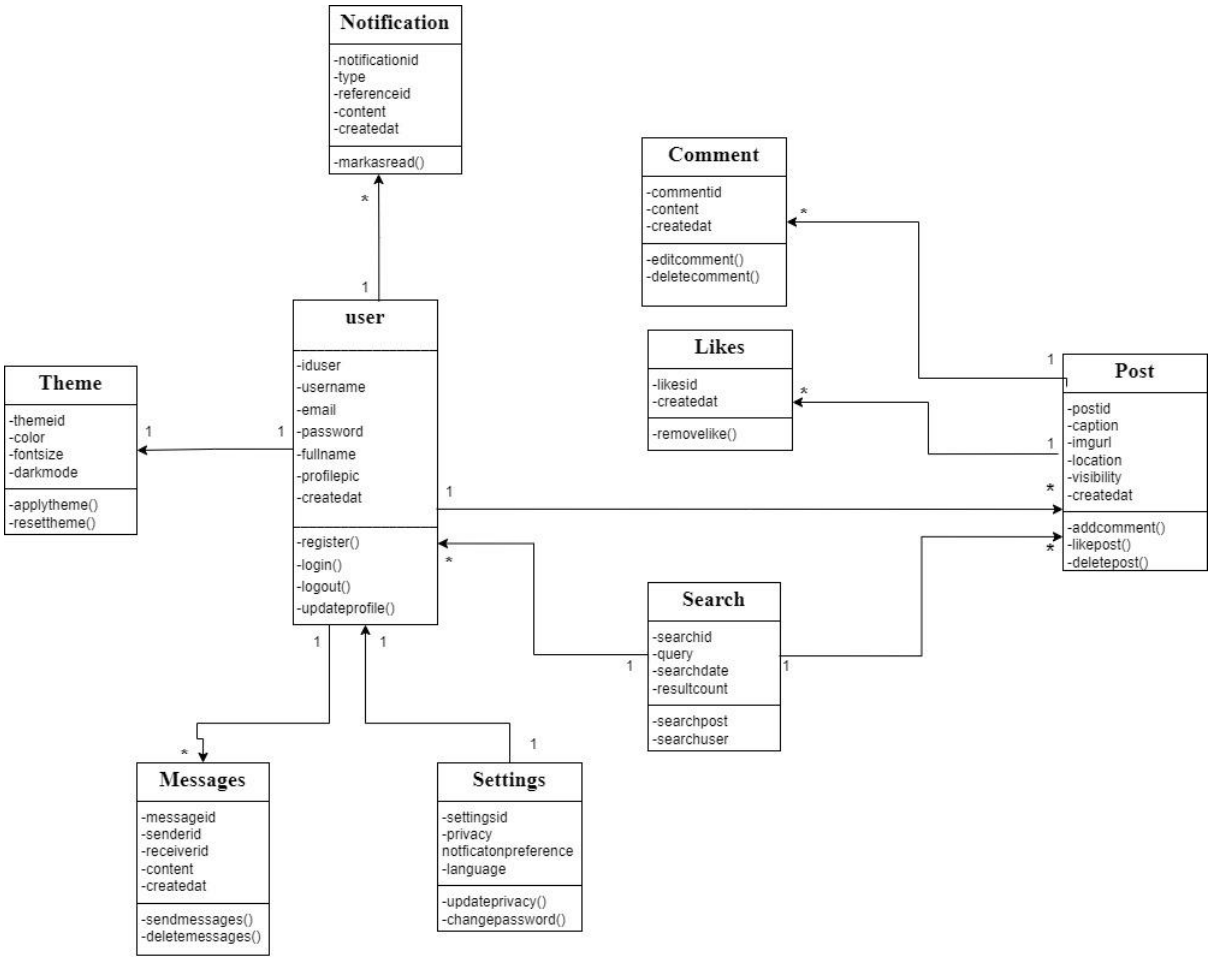
How Component Work Together:

- Client (browser) sends user actions (login, send message) to the Controller.
- Controller processes it and uses the Model for handling data.
- Model calls DatabaseHelper to fetch or update records in the SQL Server Database.
- Response flows back the same way, and updated data is shown to the user.

Work Flow:

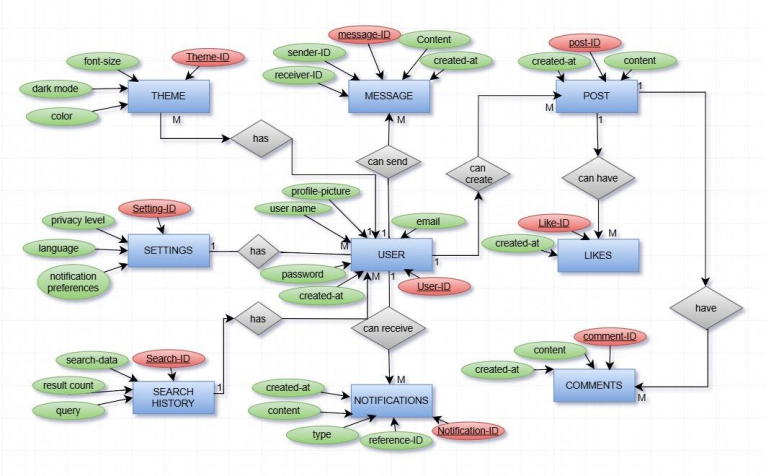


UML Class relationship diagram:



ERD Diagram:

An **Entity-Relationship Diagram (ERD)** for the **InstaVibe app** shows how data is organized and how different parts of the app (like users, posts, comments, and likes) are connected in the database.



3.4 Data Design

The system uses a SQL Server database to store key entities like Users, Posts, Messages, and Comments. Each entity is stored in tables with relevant fields (e.g., userID, username). Relationships between entities are managed using foreign keys.SQL queries are used to fetch and update data as needed.

3.4.1 Data Dictionary

The Data Dictionary provides a detailed list of important system entities, attributes, and classes used in InstaVibe.It describes each data item's name, type, and purpose.

This helps developers understand what data exists, what it means, and how it is used inside the system.

• Alphabetical List of System Entities or Major Data with Types and Descriptions:

Term / Entity	Description (Type + Purpose)
Account Controller	Controller – Handles login and signup requests
Chat Controller	Controller – Manages chat sending and receiving.
Profile Controller	Controller – Manages user profile view and edit operations.
User	Model (Class) – Stores user-related data (ID, Username, Email, Password)
Message	Model (Class) – Represents individual chat messages (ID, SenderID, ReceiverID, Text)
Username	String – User's display and login name.
Email	String – User's email for login and communication.
Password	String – User's secret password (stored encrypted).
User Id	Integer – Unique identifier for each user.
Message Id	Integer – Unique identifier for each chat message.
Sender Id	User ID of the person sending the message.
Receiver Id	Integer – User ID of the person receiving the message.
Text	String – Content of the chat message.
Database Helper	Helper Class – Executes SQL queries using ADO.NET.
Users Table	SQL Table – Stores user account information.

• Structured Approach: Functions and Function Parameters:

Function Name	Parameters	Description
Get User Profile(int userId)	userId (int): The ID of the user whose profile we want.	Gets a user's profile details from the database.
Update User Profile (User Model user)	user (UserModel) : The updated user data (like name, bio, email, etc.).	Updates the user's profile in the database.

Open Sql Connection()	(No parameters)	Opens the connection to the database.
Close Sql Connection()	(No parameters)	Closes the database connection.
ExecuteReader(string query, SqlParameter[ ])	**query**: SELECT query **parameters**: Values for the query.	Reads data from the database.
ExecuteNonQuery(string query, SqlParameter[ ])	**query:** INSERT/UPDATE query **parameters**: Values for the query	Inserts or updates data in the database.

• Object-Oriented (OO) Approach: Objects, Attributes, Methods, & Method Parameters:

1.Object: User Model

Attributes (Information stored for a user):

UserId (int):

A unique number given to each user. It is used to identify the user.

Username (string):

The user's chosen login name. It should be unique.

FullName (string):

The complete name of the user (first name + last name).

Email (string):

The user's email address used for communication and login.

Bio (string):

A short description written by the user about themselves.

ProfilePictureUrl (string):

The web link (URL) where the user's profile photo is stored.

Methods (What actions the object can perform):

GetProfile (int userId):

Finds and returns all the profile information of the user based on their ID.

UpdateProfile (UserModel user):

Updates and saves the user's profile changes to the database.

3.5 User Interface Design

InstaVibe provides a simple and friendly interface where users can manage their profiles easily. After logging in, the user is taken to their profile page. The interface is clean, mobile-responsive, and easy to navigate with all important actions available in a few clicks.

Main Features Available to the User:

View Profile: Users can see their profile information such as:

- Full name
- Username
- Email
- Bio (short intro)
- Profile picture

Edit Profile: A visible “Edit Profile” button is available on the profile page.

When clicked, an edit form opens where users can update:

- Full name
- Username
- Email
- Bio
- Profile picture (via image URL)

Save Changes: When a user makes changes and clicks “Save,” the app checks the changes, updates the record if it's fine, and shows the new info on the screen.

3.6 Design Decisions

While building **InstaVibe**, we made some important design choices to keep the system simple, easy to manage, and user-friendly.

1.Object-Oriented (OO) Design Pattern: We used an Object-Oriented approach by creating classes like UserModel and DatabaseHelper.

1. Easy to organize the code (separate user data and database operations).
2. Easy to maintain and update later.
3. Code becomes reusable and cleaner

2.Database Normalization: We normalized the database up to Second Normal Form (2NF).

- Avoids duplicate data (no repeating user details).
- Makes updates faster and reduces errors.
- Simple structure because we only manage user profiles for now.

3.Algorithm/Query Approach: We wrote simple SQL queries (like SELECT, UPDATE, INSERT) without complex algorithms

- The project’s needs are simple (read and update user data).
- Focus is on fast and clean database operations.
- Avoids unnecessary complexity.

4. Validation Technique: Client-side validation with jQuery + Bootstrap classes and server-side validation with C#.

- Client-side validation gives instant feedback (good user experience).
- Server-side validation protects the database from bad data or hackers.

5. Feedback/Alerts: We used SweetAlert2 and Bootstrap alerts.

Gives clean, modern, and easy-to-read feedback to users.

3.7 Summary

In this chapter, we explained the key design ideas of the **InstaVibe** system. We used an Object-Oriented approach to manage user data and database operations easily. The database was normalized to avoid duplicate data and keep it simple. Simple SQL queries were used for profile actions like viewing and updating profiles. Validation was added on both client-side and server-side to make the system user-friendly and secure. We also created diagrams and use cases to better explain the system flow and user interactions. All these design choices help to meet the project goal: building a clean, easy-to-use profile management system.

Chapter 4:

System Development

4.1. Architectural Interface:

An Architectural Interface defines the points of interaction between different system components or modules. This ensures compatibility and efficient integration within the overall system architecture.

4.1.1. Login:

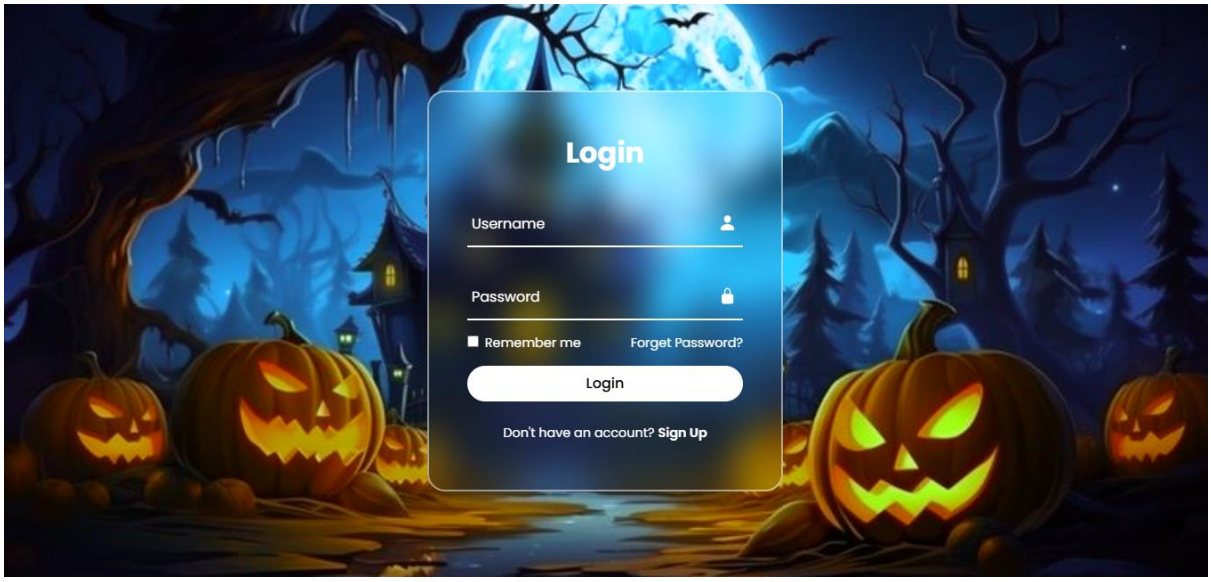


Fig 1.9 Login

4.1.2. Sign -up:

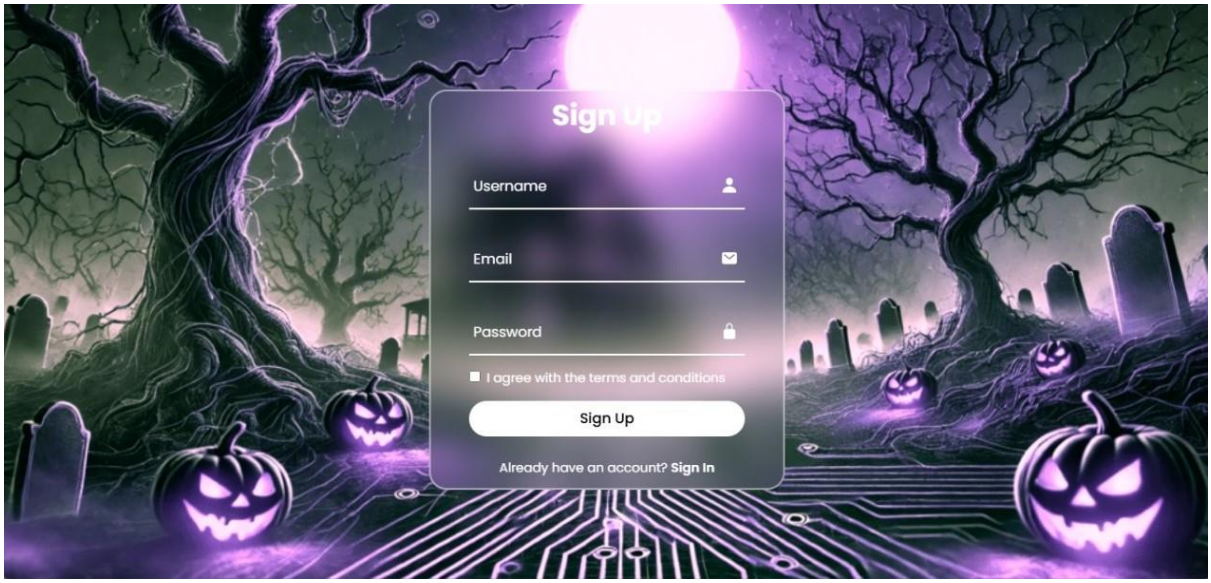


Fig 1.10 Sign -up

4.1.3. Home page:

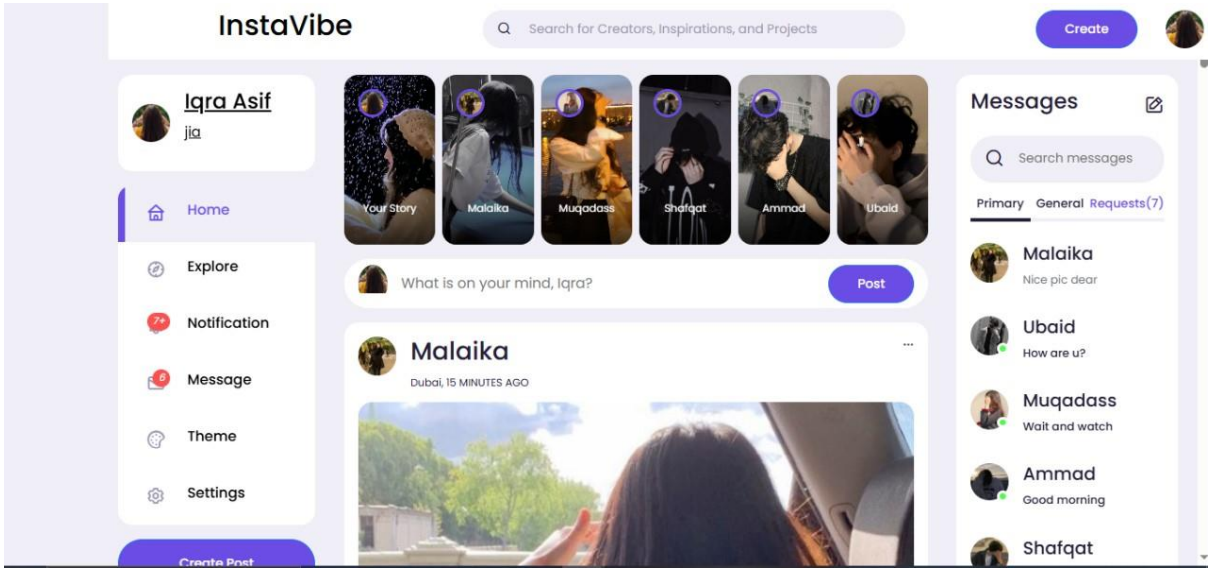


Fig 1.11 Home page

4.2 Architectural Design

Architectural Design represents the structure of data and program components required to build a computer-based system. The architectural style used for our system is a Layered Architecture, where different modules operate at separate layers.

For our social media app, the system consists of four layers, described as follows:

1. Presentation Layer (Frontend)

1. Handles user interaction and UI/UX.

2. Built using HTML, CSS, JavaScript, and optionally a frontend framework like .net framework .

3. Users can log in, upload media, like/comment on posts, send messages, and customize themes.

2. Business Logic Layer (Backend)

1. Implements the core functionalities and processes requests.

2. Developed using C# (.Net framework).

3. Manages user authentication, follow/unfollow actions, media uploads, messaging, and settings updates.

3. Data Access Layer

1. Facilitates interaction between the backend and database.

2. Handles database queries, storage, and retrieval of user data, media files, messages, and reports.

3. Using MVC 5 with ADO.NET

4. Database Layer

1. Stores all persistent data such as user accounts, media files, comments, likes, messages, and reports.

2. Ensures data consistency and security.

3. Uses MySQL server management as the primary database system.