**CHAPTER 1**

**INTRODUCTION**

**1.1 Background**

Social media has become an integral part of modern communication, with platforms like Instagram revolutionizing how people connect, share, and consume content. Instagram, known for its emphasis on photo and video sharing, has garnered over a billion active users worldwide. This proliferation underscores the importance of understanding the underlying technologies that power such influential applications.

For computer science and information technology students, building a clone of a popular application like Instagram provides a valuable hands-on experience that encompasses front-end and back-end development, user interface design, and database management. Through this process, students can gain insights into the intricacies of designing intuitive user interfaces that enhance the user experience, handling large volumes of data efficiently, and ensuring robust security measures to protect user information.

Moreover, creating an Instagram clone allows students to delve into the nuances of implementing features such as real-time notifications, multimedia processing, and social networking capabilities. They can explore how to optimize application performance for different devices and network conditions, and learn the importance of scalability to accommodate a growing user base. Additionally, students can gain practical knowledge in deploying applications on cloud platforms, understanding load balancing, and maintaining uptime.

The exercise of building such a clone also helps students familiarize themselves with the latest frameworks and libraries used in modern web and mobile development. By working on a project that mirrors a real-world application, they can better prepare for future careers in tech, gaining experience that is directly applicable to industry standards and practices. Overall, the endeavor not only enhances technical skills but also fosters a deeper appreciation for the complexities involved in creating and maintaining a globally popular social media platform like Instagram.

**1.2 Project Overview**

The Instagram Clone project is an academic endeavor aimed at replicating the core functionalities of Instagram using HTML, CSS, and Django. This project serves as a comprehensive exercise in full-stack development, covering the intricacies of web design, server-side logic, and database interaction. By developing this clone, students will gain practical insights into creating a scalable and interactive web application.

The project begins with the front-end development, where students will use HTML and CSS to design a user interface that mirrors Instagram's clean and responsive layout. This phase emphasizes the importance of creating visually appealing and user-friendly designs that adapt to various devices and screen sizes. Students will learn to implement essential UI elements such as the navigation bar, user profiles, image feeds, and interactive buttons.

Following the front-end development, the project shifts focus to the back-end, utilizing Django to handle server-side operations. Here, students will delve into the creation of robust models to represent users, posts, comments, and likes. They will explore Django's powerful ORM (Object-Relational Mapping) to efficiently interact with the database, ensuring data integrity and seamless retrieval. This phase also includes implementing authentication systems, allowing users to sign up, log in, and manage their profiles securely.

Additionally, the project incorporates real-time features such as notifications and live updates, enhancing the application's interactivity. Students will learn to use Django Channels to manage WebSockets, facilitating real-time communication between the server and the client. By integrating these dynamic features, students will understand the complexities of building responsive and engaging web applications.

Overall, the Instagram Clone project not only reinforces theoretical knowledge but also provides hands-on experience with industry-relevant technologies and practices. It prepares students for real-world scenarios, equipping them with the skills necessary to develop and maintain sophisticated web applications.

**1.3 Objective**

The primary objectives of this project are as follows:

To understand and implement the fundamental concepts of web development using HTML and CSS for the front-end. This includes mastering the structure and styling of web pages to create visually appealing and user-friendly interfaces. Additionally, it involves learning about responsive design to ensure the website functions well on various devices and screen sizes.

To develop a robust back-end using Django, a high-level Python web framework. This encompasses building a scalable and secure server-side application that can handle user authentication, database interactions, and business logic. The project also aims to incorporate best practices for software development, such as version control, testing, and deployment processes.

**1.4 Key Features**

Creating an Instagram Clone includes a number of features, ensuring a comprehensive and engaging user experience:

**User Authentication**:

Secure user registration: Allows new users to create accounts with unique usernames and passwords.

Login: Secure login process with options for email or username and password.

Logout functionalities: Ensures users can safely log out of their accounts.

Password recovery: Users can reset their password via email verification.

Social media login: Users can log in using their existing social media accounts like Facebook or Google.

**Photo Uploading and Sharing**:

Upload photos: Users can upload photos from their devices.

Add captions: Users can add descriptive or fun captions to their photos.

Share on feed: Photos can be shared on the user’s feed for their followers to see.

Photo filters: Various filters to enhance and personalize photos.

Edit photos: Basic photo editing tools like crop, rotate, and adjust brightness.

Tagging: Users can tag other users in their photos.

Location tagging: Users can add location information to their photos.

**User Profile Management**:

Customizable profiles: Users can set a profile picture, bio, and personal information.

Follow/Unfollow: Users can follow or unfollow other users.

View profiles: Users can view other users’ profiles to see their posts and activity.

Privacy settings: Users can set their profiles to public or private.

**Social Interaction**:

Like photos: Users can like photos posted by others.

Comment on photos: Users can leave comments on photos.

Direct messaging: Private messaging system for users to communicate directly.

Notifications: Users receive notifications for likes, comments, follows, and direct messages.

**Discover and Explore**:

Search functionality: Users can search for other users, hashtags, and locations.

Explore page: A curated page of popular or trending photos and profiles.

Hashtags: Users can use and search hashtags to find related content.

**Stories**:

Temporary posts: Users can post photos and videos that disappear after 24 hours.

View stories: Users can view stories from people they follow.

Story highlights: Users can save stories to their profile highlights.

**1.5 Scope of the project**

The scope of the Instagram Clone project encompasses a comprehensive approach to web development, ensuring the creation of a fully functional and user-friendly social media platform. The detailed scope includes:

**Front-End Development**

**Interface Design**: Utilizing HTML and CSS to craft a visually appealing and intuitive user interface. The design will focus on providing a responsive layout that adapts seamlessly to various screen sizes and devices, ensuring a consistent user experience.

**User Experience (UX)**: Implementing interactive elements and navigation features that enhance user engagement and make the application easy to use. Attention will be given to aesthetics and usability, incorporating best practices in UI/UX design.

**Back-End Development**

**Server-Side Logic**: Leveraging Django, a powerful web framework, to handle all server-side operations. This includes processing user requests, managing sessions, and implementing business logic.

**User Authentication**: Developing robust authentication mechanisms to manage user sign-up, login, and secure access. This involves setting up features such as password hashing, account verification, and user roles.

**Data Processing**: Implementing efficient data processing techniques to handle user-generated content, such as posts, comments, and likes, ensuring fast and reliable performance.

**Database Management**

**Database Design**: Designing a relational database using Django's Object-Relational Mapping (ORM) system. The database will be structured to store user information, posts, interactions, and metadata efficiently.

**Data Integrity**: Ensuring data integrity and consistency through the implementation of appropriate database constraints and relationships, such as foreign keys and unique constraints.

**Scalability**: Planning for scalability to accommodate growing amounts of data and users. This includes optimizing database queries and indexing critical fields to maintain performance as the application scales.

**Integration**

**Front-End and Back-End Integration**: Ensuring seamless integration between the front-end and back-end components. This involves setting up APIs and endpoints that allow smooth communication between the user interface and the server.

**Real-Time Features**: Integrating real-time functionalities, such as live notifications and instant updates to reflect interactions like comments and likes, enhancing the dynamic nature of the application.

**Third-Party Services**: Incorporating third-party services and APIs where necessary, such as cloud storage for media files, social media sharing options, and email services for notifications.

**Testing and Deployment**

**Thorough Testing**: Conducting comprehensive testing to ensure the application's functionality, performance, and security. This includes unit tests, integration tests, and end-to-end testing.

**User Feedback**: Gathering user feedback during the testing phase to identify potential improvements and address any usability issues.

**Deployment**: Deploying the application on a reliable web server, ensuring it is accessible for demonstration and use. This includes setting up a deployment pipeline, managing server resources, and configuring domain and SSL certificates for secure access.

**Maintenance**: Planning for ongoing maintenance and updates post-deployment to fix bugs, improve features, and adapt to user needs.

**CHAPTER 2**

**PROBLEM IDENTIFICATION & FEASIBILITY STUDY**

**2.1 Problem Identification**

In today's digital age, social media platforms like Instagram have become integral to communication, content sharing, and community building. These platforms are not only used for personal interaction but have also become essential tools for businesses, influencers, and content creators to reach their audiences. However, there is a significant educational need for computer science students to understand the intricacies of developing a full-fledged web application similar to Instagram.

This project aims to provide students with hands-on experience in web development using HTML, CSS, and Django, which are fundamental technologies in the modern web development stack. By engaging in this project, students will gain practical skills in front-end and back-end development, database management, user authentication, and responsive design. They will also learn about the deployment of web applications, ensuring they are prepared for the complexities of real-world web development.

The primary problem identified is the lack of practical projects in the curriculum that mimic real-world applications. This deficiency limits students' exposure to contemporary development practices and technologies, which are crucial for their future careers. Traditional coursework often focuses on theoretical aspects and small-scale projects that do not fully encapsulate the challenges and scale of real-world applications. By working on a project that emulates a platform as complex and widely-used as Instagram, students can bridge the gap between theoretical knowledge and practical application.

Moreover, this project will encourage students to think critically about user experience (UX) and user interface (UI) design, security concerns, and scalability issues. They will encounter and resolve problems such as data handling, image storage and processing, and implementing features like likes, comments, and user profiles. Through this comprehensive project, students will also develop teamwork and project management skills, as they collaborate on various aspects of the application.

In conclusion, integrating such practical projects into the curriculum is essential for providing students with a robust educational experience. It equips them with the necessary skills and knowledge to succeed in the ever-evolving field of web development. By developing an Instagram-like web application, students will not only learn the technical aspects of web development but also gain insights into the importance of user engagement, community building, and the impact of social media on contemporary society.

**2.2 Feasibility Study**

**2.2.1 Technical Feasibility**

Technical feasibility assesses whether the technology and resources available are capable of meeting the requirements of the project. This involves evaluating the capabilities of current hardware, software, and technical expertise to determine if they can support the project's goals effectively. For designing the user interface and ensuring a responsive design, it is crucial to select a high-level Python web framework. Such frameworks, like Django or Flask, encourage rapid development and clean, pragmatic design, which helps streamline the development process and produce a user-friendly, efficient interface.

In addition to the front-end considerations, choosing the right database management system (DBMS) is vital for the back-end. The DBMS must be capable of efficiently storing and managing user data, posts, comments, and likes. This includes ensuring data integrity, security, and scalability to handle large volumes of data and high traffic. Options such as PostgreSQL or MySQL for relational databases, or MongoDB for a NoSQL approach, should be evaluated based on the specific needs of the application.

Furthermore, the technical feasibility study should encompass the integration of these systems with existing infrastructure, potential technological risks, and the availability of skilled personnel to support the project. This comprehensive evaluation helps identify any potential obstacles early on and develop strategies to address them, ensuring a smoother development process and a higher likelihood of project success. By thoroughly assessing these technical aspects, the project team can make informed decisions that align with the project's goals and resource constraints.

**2.2.2 Operational Feasibility**

Operational feasibility examines the degree to which the required tasks and procedures can be implemented effectively within the existing organizational environment. This includes evaluating whether the current operational processes and systems can support the new project and if the staff has the capability to handle the additional workload.

One of the critical components of operational feasibility for a social media application is the user registration and authentication system. This system must be robust to ensure secure and efficient user onboarding while protecting against unauthorized access.

Profile management is another essential feature, allowing users to upload profile pictures, update personal information, and customize their profiles to reflect their identities.

Post creation and sharing functionality is at the core of the application, enabling users to create posts that include images and captions. This feature should be intuitive and user-friendly, encouraging frequent use and engagement.

Interaction features, such as liking and commenting on posts, are vital for fostering community and user engagement. These features should be designed to be responsive and seamless, providing users with a smooth experience.

The ability to follow and unfollow users is crucial for personalizing the user experience, allowing users to curate their feeds based on their interests.

A feed that displays posts from followed users is essential for keeping users engaged and updated with content that is relevant to them. This feed should be dynamically updated and designed to prioritize content based on user interactions and preferences.

Additionally, the operational feasibility study should assess the training needs of the staff, the compatibility of the new system with existing workflows, and the potential need for additional resources or changes in organizational structure. By thoroughly examining these factors, the project team can ensure that the system will be effectively integrated and supported within the current operational framework, leading to a successful implementation.

**2.2.3 Economic Feasibility**

Economic feasibility evaluates whether the project is financially viable and if the benefits outweigh the costs. This involves a thorough analysis of the costs associated with the project against the expected benefits to determine if the project is a sound investment.

**Development Costs**: For this project, development costs are minimal because the primary resource being utilized is student time and effort. Since this is a college project, the students are not being paid salaries, which significantly reduces the overall expense. The main investment is the time and dedication of the students working on the project, which is part of their educational experience.

**Software Costs**: All required software for the project, including Django for the web framework and various database systems, are open-source and free. This eliminates the need for purchasing expensive software licenses, making the project more economically feasible. The use of open-source software not only reduces costs but also provides flexibility and community support for the development process.

**Hardware Costs**: The project can utilize existing computer lab facilities provided by the college. This means there is no need to invest in new hardware, as the current infrastructure is sufficient to support the development and testing phases. Utilizing existing resources helps keep the project within budget and ensures that it can be completed without additional financial strain.

**Maintenance Costs**: For a college project, maintenance costs are minimal. The project primarily requires occasional updates and bug fixes, which can be handled by the students or faculty overseeing the project. Since the scope of the project is likely limited to the academic environment, extensive maintenance or ongoing operational costs are not anticipated.

In addition to these costs, the economic feasibility analysis should consider any potential indirect costs, such as the opportunity cost of students’ time or any additional resources that might be required unexpectedly. It should also evaluate the potential benefits, such as the educational value to students, the enhancement of their skills, and any potential for the project to be expanded or utilized beyond the college setting.

By carefully balancing the costs against the anticipated benefits, the economic feasibility study aims to ensure that the project is not only achievable but also a worthwhile endeavor from a financial perspective. This comprehensive analysis helps in making informed decisions about proceeding with the project and ensures that resources are used effectively.

**CHAPTER 3**

**REQUIREMENT ANALYSIS**

**3.1 Introduction**

This project aims to develop a clone of Instagram, a popular social media platform, using HTML, CSS, and Django. The application will allow users to share photos and videos, follow other users, like and comment on posts, and view their feed. This project is intended to provide hands-on experience in full-stack web development and reinforce the principles of user interface design, backend development, and database management.

The development process will start with the design of a responsive and user-friendly interface using HTML and CSS. This will ensure that the application is visually appealing and accessible on various devices, including desktops, tablets, and smartphones. Emphasis will be placed on creating an intuitive layout that mirrors the user experience of Instagram, making it easy for users to navigate and interact with the platform.

On the backend, Django will be used to handle server-side operations, manage data, and implement business logic. This powerful Python web framework is known for its simplicity and flexibility, making it an excellent choice for developing complex web applications. Django's built-in features, such as the admin interface and authentication system, will be leveraged to streamline development and enhance security.

The project will also involve the integration of a robust database management system to store user information, posts, comments, and likes. This will ensure data integrity and support efficient data retrieval and manipulation. By using Django's ORM (Object-Relational Mapping), the project will facilitate seamless interaction between the application and the database, allowing for easy data management and queries.

Furthermore, users will have the ability to create profiles, upload and edit photos and videos, and engage with other users' content through likes and comments. The following functionality will enable users to build their social networks within the app, and a dynamically updated feed will display posts from followed users, ensuring that users stay engaged with fresh and relevant content.

This comprehensive project will not only enhance technical skills in web development but also provide valuable insights into the lifecycle of web application development, from initial design and development to testing and deployment. By replicating the core features of Instagram, the project will offer a deep understanding of how large-scale social media platforms operate and the various challenges involved in building and maintaining such applications.

**3.2 Functional Requirements**

**3.2.1 User Registration**

Users should be able to sign up using an email address and password, which will serve as their primary credentials for accessing the application. During the sign-up process, user information should be meticulously validated for correctness and completeness. This validation process includes checking for the proper format of the email address, ensuring the password meets security standards (such as minimum length, inclusion of special characters, and avoidance of common passwords), and verifying that all required fields are filled out accurately.

To further enhance security and ensure the validity of the provided email address, an email verification process should be implemented. After the initial registration, an automated email containing a unique verification link should be sent to the user's email address. The user must click this link to activate their account, which confirms that the email address is valid and accessible by the user. This step helps to prevent fake or mistyped email addresses from being used and adds an additional layer of security to the sign-up process.

Moreover, the system should be designed to handle edge cases, such as users attempting to sign up with previously registered email addresses, ensuring appropriate error messages are displayed. Password recovery options should also be available, allowing users to reset their passwords via email verification if they forget their credentials. These features collectively contribute to a secure and user-friendly registration system, ensuring that only legitimate users gain access to the platform while maintaining the integrity of the user data.

**3.2.2 User Login**

Users should be able to log in using their email address and password. This requires implementing a secure authentication system that verifies user credentials and ensures that only authorized users can access their accounts. The login process should include measures such as encryption of passwords, protection against brute force attacks, and validation of email addresses.

Implementing session management is crucial to maintain user state across different pages. This involves creating sessions when users log in, which keep track of their authenticated state and other relevant information. Sessions should be securely managed, with unique session IDs assigned to each user and stored in a secure manner, typically in cookies. These sessions should have a timeout feature to log users out after a period of inactivity, enhancing security.

Moreover, session management should support features like 'remember me,' allowing users to stay logged in across browser sessions, and secure logout processes to ensure sessions are properly terminated. Proper session handling also includes protection against session hijacking and fixation by employing techniques such as regenerating session IDs after login and using secure, HTTP-only cookies.

Overall, the authentication and session management system should provide a seamless yet secure user experience, ensuring that users can log in, stay logged in, and securely log out, all while maintaining their state across different pages within the application.

**3.2.3 User profile**

Users should have a personal profile page that serves as their digital identity hub within the platform, showcasing not only their personal information but also their contributions and interactions. This profile page acts as a virtual reflection of the user's personality, interests, and activities on the platform. It should be designed to be visually appealing, intuitive to navigate, and highly customizable to allow users to express themselves authentically.

In addition to displaying basic information such as username, profile picture, and bio, users should have the flexibility to edit and update various aspects of their profile. This includes not only changing their profile picture but also refining their bio, updating personal details like contact information or interests, and even customizing the overall aesthetic of their profile page to reflect their unique style and preferences. Providing users with the tools to curate their online persona enhances their sense of ownership and agency over their digital presence.

Furthermore, users should have easy access to insights and analytics about their profile activity, such as the engagement metrics of their posts, the demographics of their followers, and trends in their interactions over time. These insights empower users to better understand their audience, refine their content strategy, and foster meaningful connections within the platform community.

On the social front, users should be able to effortlessly navigate their network of connections by viewing their list of followers and the users they are following. This feature not only facilitates social discovery and relationship management but also encourages reciprocity and engagement within the user community. Additionally, incorporating features like suggested users to follow or mutual connections can further enrich the networking experience and expand users' social circles.

Overall, the personal profile page should not only serve as a static repository of information but also as a dynamic hub for self-expression, interaction, and community engagement. By prioritizing user empowerment, personalization, and social connectivity, the platform can foster a vibrant and inclusive digital ecosystem where users feel valued, connected, and inspired to share their voice with the world.

**3.3 Post Management**

**3.3.1 Create Post**

Authenticated users should be able to create new posts with photos or videos, enriching the platform with diverse content and enabling users to express themselves creatively. This feature serves as the cornerstone of user engagement, facilitating the sharing of experiences, ideas, and moments.

Implementing image and video upload functionality with proper validation and storage is crucial to ensure the integrity and security of the platform. This involves incorporating robust validation mechanisms to verify the file types, size, and content to prevent malicious uploads and ensure compliance with platform guidelines. Additionally, the storage infrastructure should be scalable and reliable, capable of handling large volumes of media files efficiently while maintaining high performance and availability.

Users should be able to add captions to their posts, providing context and enhancing the storytelling aspect of the content. Captions enable users to convey additional information, emotions, or messages associated with their posts, making them more engaging and meaningful to the audience.

Furthermore, providing features like image cropping, filters, and editing tools can empower users to enhance their posts and personalize their content according to their preferences. This not only improves the visual appeal of the content but also encourages users to actively participate and contribute to the platform.

Integration with third-party services for media hosting and processing can further enhance the functionality and capabilities of the platform, enabling advanced features such as real-time video transcoding, content moderation, and analytics.

Overall, prioritizing user-friendly design, seamless functionality, and robust security measures in the implementation of media upload and post creation features is essential to delivering a compelling and reliable user experience.

**3.3.2 View Post**

Posts play a central role in driving user engagement and content discovery within a social media platform. They serve as the primary means for users to share their thoughts, experiences, and media with their audience. Therefore, ensuring posts are prominently displayed across various sections of the platform is crucial for maximizing user interaction and content visibility.

Posts should be displayed on the user's feed, profile, and search results to ensure they reach the widest possible audience. The feed is where users encounter a curated selection of content from accounts they follow, providing them with a personalized stream of updates and activities. In contrast, the profile serves as a comprehensive repository of a user's own posts, allowing visitors to explore their content history and interests. Integrating posts into search results enhances content discoverability, enabling users to find relevant posts based on keywords, hashtags, or user profiles.

Each post should include essential components such as the content itself, user information, timestamp, caption, and interaction buttons. The content may comprise text, images, videos, or a combination thereof, reflecting the diverse range of content types users share. User information, including the profile picture and username, adds context to the post and facilitates user identification. The timestamp indicates when the post was published, providing users with temporal context and aiding in content freshness assessment. Captions provide additional context or commentary to accompany the post, enriching the user experience and encouraging engagement.

Interaction buttons, such as 'like' and 'comment,' are essential for facilitating user engagement with posts. These buttons allow users to express their appreciation for content, provide feedback, or initiate conversations with the post creator and other users. Additionally, features like 'share' or 'save' can further enhance post interaction and dissemination, enabling users to amplify content reach and revisit favorite posts later.

Furthermore, incorporating features like post tagging, categorization, or content filtering options can enhance post organization and navigation, facilitating content discovery and user engagement. Implementing algorithms to personalize post recommendations based on user preferences and behavior can also enhance the relevance and engagement of displayed posts.

In summary, optimizing the display and interaction features of posts across different sections of the platform is essential for fostering user engagement, content discovery, and community interaction within the social media ecosystem.

**3.3.3 Like and Comment**

Users should indeed be able to interact with posts in various ways to foster engagement and communication within the platform. Liking and unliking posts is a fundamental feature that allows users to express their appreciation for content they enjoy. It's a simple yet powerful way to signal approval and encourage content creators.

Moreover, providing users with the ability to add, edit, and delete comments on posts adds depth to the interaction. This not only facilitates discussions but also allows users to correct mistakes or update their thoughts over time. However, ensuring proper moderation tools and guidelines is crucial to maintain a positive and constructive atmosphere within the community.

Displaying the number of likes and comments for each post serves multiple purposes. Firstly, it provides social proof, indicating the popularity and engagement level of the post. This can influence other users to engage with the content as well. Additionally, it offers valuable feedback to content creators, helping them understand their audience and tailor their future posts accordingly.

To further enrich the user experience, incorporating features like real-time notifications for new likes and comments can enhance user engagement and retention. Additionally, allowing users to bookmark or save posts they find interesting for later viewing can encourage prolonged interaction with the platform.

Furthermore, implementing measures to prevent abuse, such as spamming or artificially inflating likes, is essential to maintain the integrity of the platform. This may include algorithms to detect suspicious activity, as well as community-driven reporting and moderation systems.

Overall, the ability for users to interact with posts through likes and comments, accompanied by transparent metrics displaying engagement levels, is key to fostering a vibrant and engaging community within the platform.

**3.3.4 News Feed**

The news feed should be the heart of the social media experience, providing users with a curated stream of content from the accounts they follow. Implementing a chronological order for posts ensures that users see the latest updates from their network as they happen, promoting real-time engagement and fostering a sense of immediacy.

Users should be able to scroll seamlessly through the feed to view more posts, with infinite scrolling or pagination options available to accommodate different browsing preferences. This continuous flow of content encourages users to explore further and spend more time engaging with the platform.

In addition to displaying posts from followed users, the news feed could incorporate algorithms to highlight particularly relevant or popular content, enhancing the overall user experience. Features like trending topics or personalized recommendations based on user interests and engagement history can add depth and variety to the feed, keeping users engaged and coming back for more.

Furthermore, integrating interactive elements within the feed, such as the ability to like, comment, and share posts directly from the feed itself, enhances user engagement and encourages meaningful interactions. Users should also have the option to filter or customize their feed based on preferences, such as sorting by content type or prioritizing posts from specific users or topics.

Overall, designing a dynamic and user-friendly news feed is crucial for keeping users engaged and satisfied with the platform. By prioritizing content relevance, ease of navigation, and interactive features, the news feed becomes a powerful tool for fostering connections, sparking conversations, and driving user retention.

**CHAPTER 4**

**PROJECT PLANNING AND SCHEDULING**

**4.1 Pert Chart:**

A PERT chart is a project management tools used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique. A PERT chart presents a graphic illustration of a project as network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labelled vectors (directional lines) representing tasks in the project.

The direction of the arrows on the lines indicates the sequence of tasks.

**Programming**

**SRS And Design**

**Alpha Testing**

**Beta Testing**

10

8

5

3

25 20 30 15 15

1

2

11

9

7

6

4

10 10 120 30 5

**User Requirement and Analysis**

**Training**

**User Test**

**Writing Manuals**

**Designing**

Fig 4.1 Pert Chart

**CHAPTER 5**

**HARDWARE & SOFTWARE SPECIFICATION**

**5.1 Hardware Specification**

This document outlines the hardware and software requirements for developing an Instagram clone using HTML, CSS, and Django as part of a college project. The specifications ensure that the development, testing, and deployment processes are conducted efficiently. The recommended hardware specifications are as follows:

**Server:**

Processor: Intel Core i5 or equivalent

RAM: 8 GB or higher

Storage: 256 GB SSD or higher

**Database Server:**

Processor: Intel Core i5 or equivalent

RAM: 8 GB or higher

Storage: 256 GB SSD or higher

Network Interface: Gigabit Ethernet

**Client Machines:**

Processor: Intel Core i3 or equivalent

RAM: 4 GB or higher

Storage: 128 GB SSD or higher

Network Interface: 100 Mbps Ethernet or Wi-Fi

**5.2 Software Specification**

The Instagram clone will be developed using a combination of server-side and client-side technologies. The software specifications include:

**Server-Side Technologies:**

Operating System: Windows Server 2016 or later

Web Server: Apache 2.4

Database Management System: MySQL 5.7 or later

Server-Side Scripting Language: PHP 7.4 or later

**Client-Side Technologies:**

Web Browser: Latest versions of Chrome, Firefox, Safari, or Edge

Client-Side Scripting: JavaScript, jQuery

**Development Tools:**

XAMPP: Version 8.0.9 or later for local development and testing

Integrated Development Environment (IDE): Visual Studio Code or any preferred PHP IDE

**Version Control:**

Git: Version control for collaborative development

**Security:**

SSL/TLS: Ensure secure data transmission over the network

Firewall: Implement firewall rules to restrict unauthorized access

Anti-malware Software: Regularly updated anti-malware software on server and client machines

**CHAPTER 6**

**CHOICE OF TOOLS & TECHNOLOGY**

**6.1 Django Framework**

Django, chosen as the primary backend framework, brings a wealth of advantages to the project. Its reputation for facilitating rapid development and maintaining clean, pragmatic design principles makes it an ideal choice for efficiently building the server-side infrastructure.

With Django handling server-side logic, developers can focus on implementing core features and functionalities without getting bogged down in repetitive tasks. Its extensive ecosystem of packages and libraries further streamlines development, offering solutions for common tasks like user authentication, data validation, and content management.

Scalability is another key strength of Django. As the application grows and user demand increases, Django's scalability features, such as support for distributed architectures and seamless integration with caching solutions like Redis or Memcached, ensure that the backend can handle high volumes of traffic without sacrificing performance.

Furthermore, Django's built-in security features provide robust protection against common web vulnerabilities, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). Its authentication and authorization system offers granular control over user permissions, ensuring that sensitive data remains secure and access is restricted to authorized users only.

In addition to serving as the backbone for server-side logic and database interactions, Django will also power the REST API for client-server communication. This API will enable seamless data exchange between the frontend and backend, allowing for dynamic updates and real-time interactions within the application.

Overall, Django's combination of rapid development capabilities, scalability, security features, and support for building RESTful APIs makes it a robust choice for powering the backend of the social media platform. By leveraging Django's strengths, the development team can create a reliable and performant backend infrastructure that forms the foundation for a successful and scalable application.

**6.2 Python**

Python's selection as the primary programming language for backend development brings numerous advantages to the project. Python's readability, simplicity, and extensive library ecosystem make it an ideal choice for rapid development and efficient maintenance of the backend infrastructure.

Django, a high-level Python web framework, complements Python's strengths by providing a robust set of tools and conventions for building web applications. Leveraging Django's built-in features, such as its powerful ORM (Object-Relational Mapping) system, authentication mechanisms, and URL routing capabilities, streamlines the development process and accelerates time-to-market.

Moreover, Python's popularity within the developer community ensures access to a vast pool of talent and resources. Developers proficient in Python are readily available, reducing hiring challenges and facilitating knowledge sharing and collaboration.

In addition to Django, Python's versatility extends to other backend frameworks and technologies, such as Flask and FastAPI, offering flexibility in choosing the right tools for specific project requirements. This adaptability allows the team to tailor the backend architecture to suit the project's scalability, performance, and security needs effectively.

Furthermore, Python's thriving ecosystem of third-party libraries and packages enhances productivity by providing pre-built solutions for common tasks, ranging from database management to API integration and beyond. By leveraging these libraries, developers can focus on implementing core business logic and innovative features, rather than reinventing the wheel.

Overall, Python's combination of readability, simplicity, extensive frameworks, and rich ecosystem makes it an excellent choice for backend development, empowering the team to build robust, scalable, and maintainable solutions that meet the project's objectives effectively.

**6.3 Data Flow Diagram**

The data flow diagram (DFD) is an invaluable tool utilized during the design phase of software engineering projects. Developed by Larry Constantine, it offers a graphical representation of how data moves within a system, illustrating the processes involved and the interactions between various components. Often referred to as a "bubble chart," the DFD visually maps out the flow of information, helping developers and stakeholders alike to comprehend the system's architecture and functionality.

At its core, the DFD serves as a blueprint for understanding data transformation and processing within a system. It breaks down complex processes into simpler, more digestible components, making it easier to identify potential bottlenecks, inefficiencies, or areas for optimization. By visualizing the flow of data from input to output, the DFD facilitates communication among team members and stakeholders, ensuring everyone shares a common understanding of how the system operates.

One of the key strengths of the DFD lies in its versatility and scalability. It can be adapted to represent systems of varying sizes and complexities, from simple data processing workflows to intricate enterprise architectures. Moreover, the DFD can evolve alongside the project, serving as a dynamic tool for documenting changes and updates over time.

In addition to aiding in the design phase, the DFD plays a crucial role in requirements analysis, system documentation, and troubleshooting. It helps identify data sources, sinks, and stores, clarifying the flow of information and highlighting dependencies between different components. This, in turn, informs decision-making processes and guides the development of robust, efficient software solutions.

Overall, the data flow diagram stands as a cornerstone of software engineering, offering clarity, insight, and structure to the design and development process. Its visual representation of data flow fosters collaboration, enhances understanding, and ultimately contributes to the successful delivery of high-quality software systems. The purpose of DFD is major transformation that will become in system design symbols used in DFD: -

In the DFD, four symbols are used and they are as follows.

1. A square defines a source (originator) or destination of system data.

1. An arrow identifies data flow-data in motion. It is 2a pipeline through which information flows.

1. A circle or a “bubble “(Some people use an oval bubble) represents a process that transfers informing data flows into outgoing data flows.

1. An open rectangle is a data store-data at rest, or a temporary

repository of data.

**6.4 Context Level Diagram**

The context diagram, also known as a zero-level Data Flow Diagram (DFD), provides a bird's eye view of the entire system and its operating environment. At this level, the system is depicted as a single process, encapsulating the Instagram Clone application along with its interactions with external entities.

In this diagram, the Instagram Clone is represented as a singular entity, illustrating its role within the broader ecosystem. External entities such as users, servers, databases, and other systems are depicted as interacting with the Instagram Clone, highlighting the flow of data and information between them.

The context diagram serves as a crucial tool for understanding the system's scope and boundaries. It helps stakeholders visualize the interactions between the Instagram Clone and its environment, identifying key inputs, outputs, and interfaces. By clearly delineating the system's context, the diagram facilitates effective communication and collaboration among project team members, stakeholders, and developers.

Furthermore, the context diagram lays the foundation for developing more detailed DFDs and system models at lower levels of abstraction. It provides a starting point for decomposing the system into smaller, more manageable components, allowing for a deeper understanding of its internal processes and data flows.

Overall, the context diagram serves as a valuable reference point for system analysis, design, and development, guiding stakeholders through the complexities of the Instagram Clone application and helping ensure its successful implementation. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level DFD comes into play.

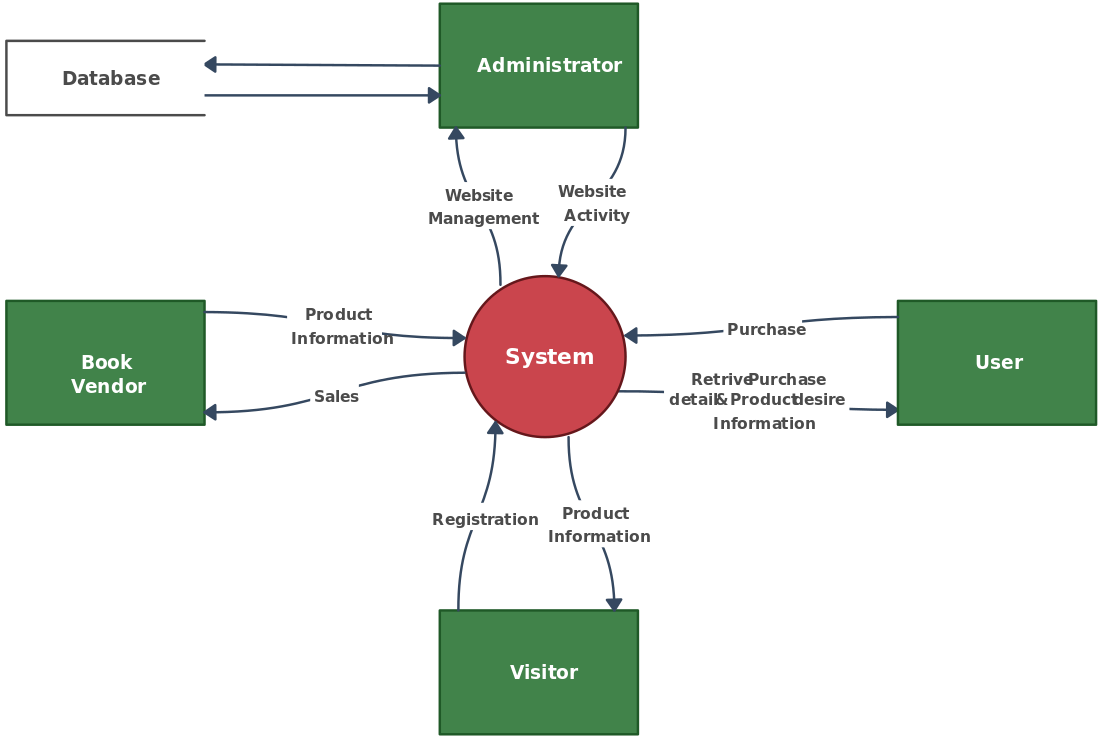
****

Fig 6.1 0-Level DFD

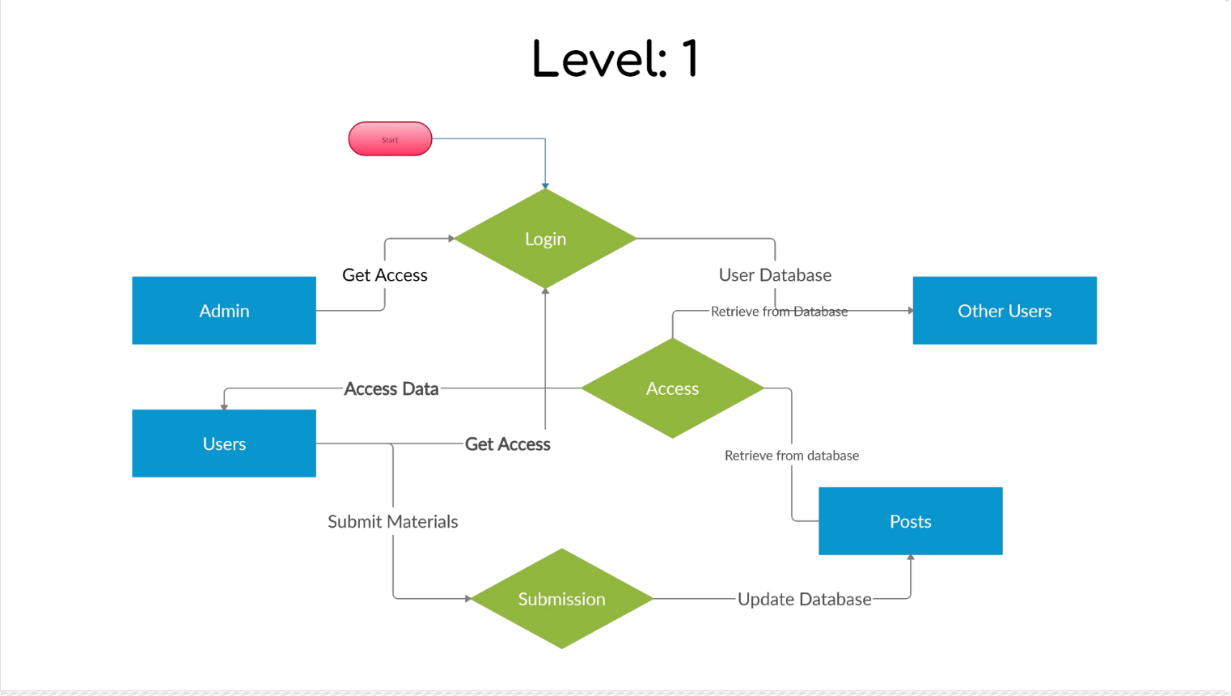
****

Fig 6.2 1st Level DFD

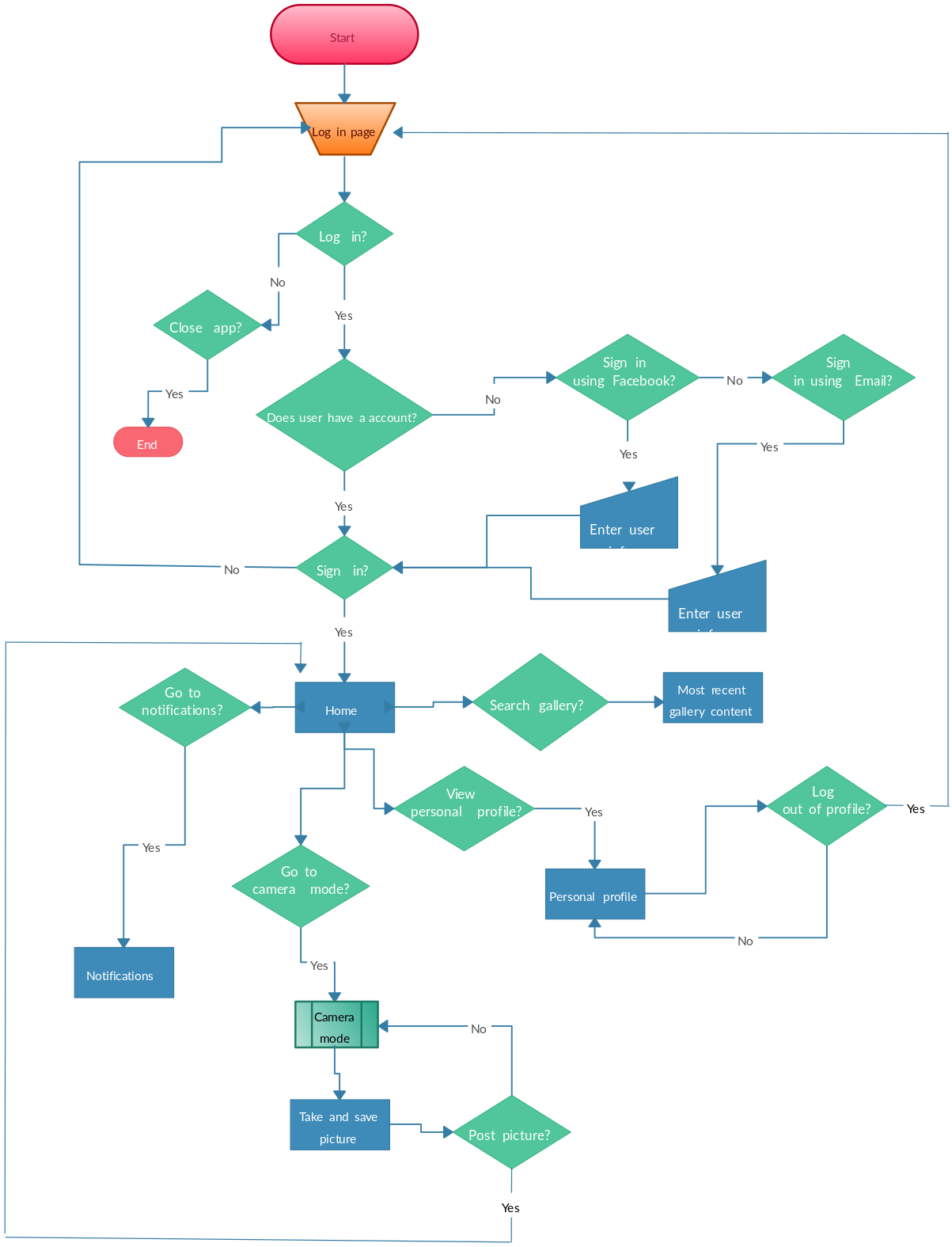


Fig 6.3 2nd Level DFD

**CHAPTER 7**

**ER-DIAGRAM**

**7.1 Entity-relationship model: -**

The entity-relationship model or entity-relationship diagram (ERD) is a data model or diagram for high-level descriptions of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity-relationship diagrams.

User (UserID, Username, Email, Password, ProfilePicture, Bio, DateJoined)

|

|----< Post (PostID, UserID, ImageURL, Caption, Timestamp)

|

|----< Comment (CommentID, PostID, UserID, Text, Timestamp)

|

|----< Like (LikeID, UserID, PostID)

|

|----< Follow (FollowID, FollowerID, FollowedID)

**7.2 Class Diagram: -**

Instagram App represents the main Django application containing the models, views, templates, and static files.

User represents the users of the system. Each user has a username, email, password, profile picture, bio, and lists of followers and following users.

Post represents the posts made by users. Each post has an author (a User), an image, a caption, a creation date, a list of users who liked the post, and a list of comments.

Comment represents the comments made on posts. Each comment has an author (a User), the post it belongs to, content, and creation date.

Like represents the likes given by users to posts. Each like is associated with a post and a user.

User

Instagram App

User

-User ID

-Username

-Password

-Role

+login ()

+logout ()

Authentication

Username

password

models.py: Django models

templates/: HTML templates

static/: CSS, JS, images

itemId: Int

author: User

image: Image

Caption: String

Likes: List[User]

Post

-User ID

-Username

-Password

+login ()

+logout ()

**CHAPTER 8**

**DATABASE**

**8.1 Admin**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Email** | **Name** | **Mobile** | **Address** | **Dob** | **Password** |
| anmol@kiet.edu | Anmol | 2147483647 | Delhi | 2023-10-03 | 1234 |

Table 8.1 Admin

In the context of a database table, the attributes "email," "name," "mobile," "Address," "Dob" (Date of Birth), and "password" represent the columns that store specific information related to individuals, likely users of a system.

**Email:**

This attribute stores the email address of an individual. It is a unique identifier and is commonly used for user authentication and communication.

**Name:**

The "name" attribute typically stores the full name of an individual. It might be divided into first name, middle name, and last name for more detailed records.

**Mobile:**

This attribute stores the mobile or phone number of an individual. It is valuable for communication purposes and might be used for sending notifications or alerts.

**Address:**

The "Address" attribute stores the physical address or location details of an individual. It could include components such as street address, city, state, and postal code.

**Dob (Date of Birth):**

The "Dob" attribute stores the date of birth of an individual. It is commonly used to determine the age of the person and might be used for age-specific functionalities or personalization.

**Password:**

The "password" attribute stores a securely hashed or encrypted version of the user's password. It is a critical attribute for user authentication, ensuring secure access to the system.

**8.2 Users**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Name** | **Mobile** | **Email** | **address** | **Dob** | **Pincode** | **password** |
| 1 | Akarsh Vishwakarma | 2147483647 | [akarsh.2224mca1164@kiet.edu](mailto:akarsh.2224mca1164@kiet.edu) | Varanasi | 2024-01-02 | 201206 | 9026 |
| 2 | Anmol | 2147483647 | [j@gmail.com](mailto:j@gmail.com) | Delhi | 2023-10-31 | 230402 | 1234677 |
| 3 | Gaurav Pandey | 2147483647 | [s2@gmail.com](mailto:s2@gmail.com) | Pratapgarh | 2024-01-01 | 230434 | 1234 |

Table 8.2 Employee

In a database table with attributes such as "Id," "Name," "Mobile," "Email," "Address," "Dob" (Date of Birth), "Pin code," and "Password," each attribute serves a specific role in capturing and organizing information about individuals.

**Id:**

The "Id" attribute is typically a unique identifier assigned to each individual in the database. It serves as a primary key, ensuring that each record can be uniquely identified and referenced.

**Name:**

The "Name" attribute stores the full name of an individual. It is a fundamental piece of personal information and is often used for identification and communication purposes.

**Mobile:**

The "Mobile" attribute stores the mobile or phone number of an individual. This information is crucial for contact and communication, facilitating outreach or notifications.

**Email:**

The "Email" attribute stores the email address of an individual. It serves as a unique identifier for user accounts and is commonly used for communication and login credentials.

**Address:**

The "Address" attribute captures the physical address or location details of an individual. It might include components such as street address, city, and state, providing a comprehensive overview of an individual's residence.

**Dob (Date of Birth):**

The "Dob" attribute records the date of birth of an individual. This information is valuable for age-related functionalities, personalized services, and can contribute to demographic analysis.

**Pin code:**

The "Pin code" attribute stores the postal code associated with an individual's address. It helps in geographically categorizing and organizing individuals based on their location.

**Password:**

The "Password" attribute stores a securely hashed or encrypted version of the user's password. It is a critical attribute for user authentication, ensuring the security of user accounts by protecting access to sensitive information.

**Id:**

The “Id” attribute typically serves as a unique identifier for each record in the database table. It is a primary key that ensures each entry can be uniquely identified and referenced.

**Contact:**

The “Contact” attribute likely stores contact information related to the users. This facilitates communication and coordination with the users.

**Mobile no:**

The "Mobile no" attribute captures the contact number of the supplier. This information is crucial for communication purposes, allowing for easy and direct contact with the users when necessary.

**Address:**

The "Address" attribute records the official address of the users. This could include details such as street address, city, and postal code, providing a comprehensive location reference.

**Resident address:**

The "Resident address" attribute, if included, may store the personal or residential address of the users. This information might be relevant for users who operate individually or as smaller entities.

**CHAPTER 9**

**FORM DESIGN**

**9.1 Login**

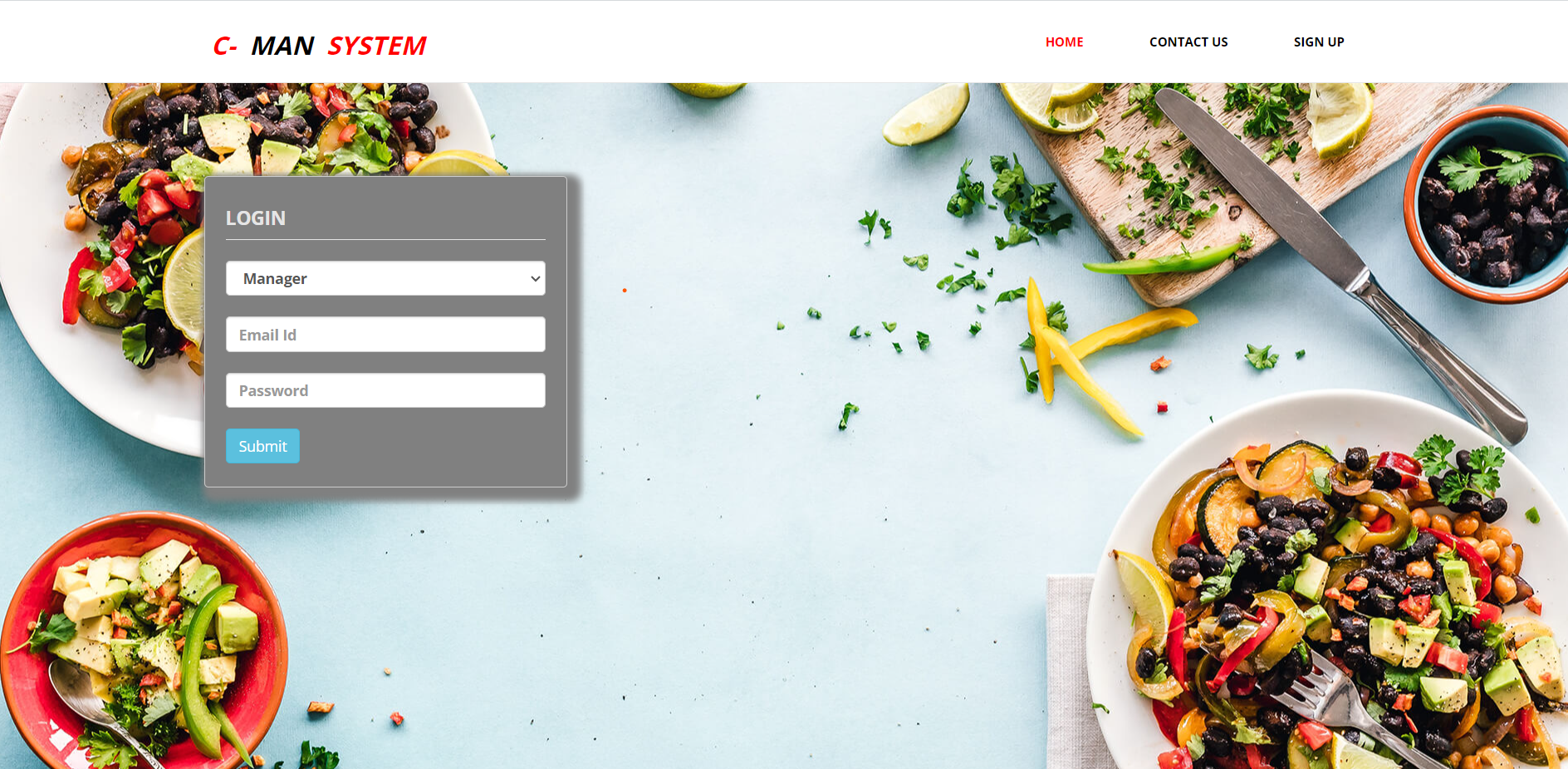
****

Fig. 9.1 Login

The login module in the Instagram Clone is a pivotal component responsible for securely authenticating users and regulating access to the system. This module verifies user identity by validating entered credentials, typically consisting of a username and password.

Security measures, including password hashing and salting, are implemented to safeguard user credentials. The login module may also include features such as an account lockout mechanism to counter multiple failed login attempts, logging and auditing for monitoring user activities, and password recovery options for forgotten passwords.

Overall, the login module is of paramount importance as it serves as the initial checkpoint, ensuring that only authorized users gain access to the Cloned while upholding security and user-friendly practices.

**9.3.1 Profile Updation**

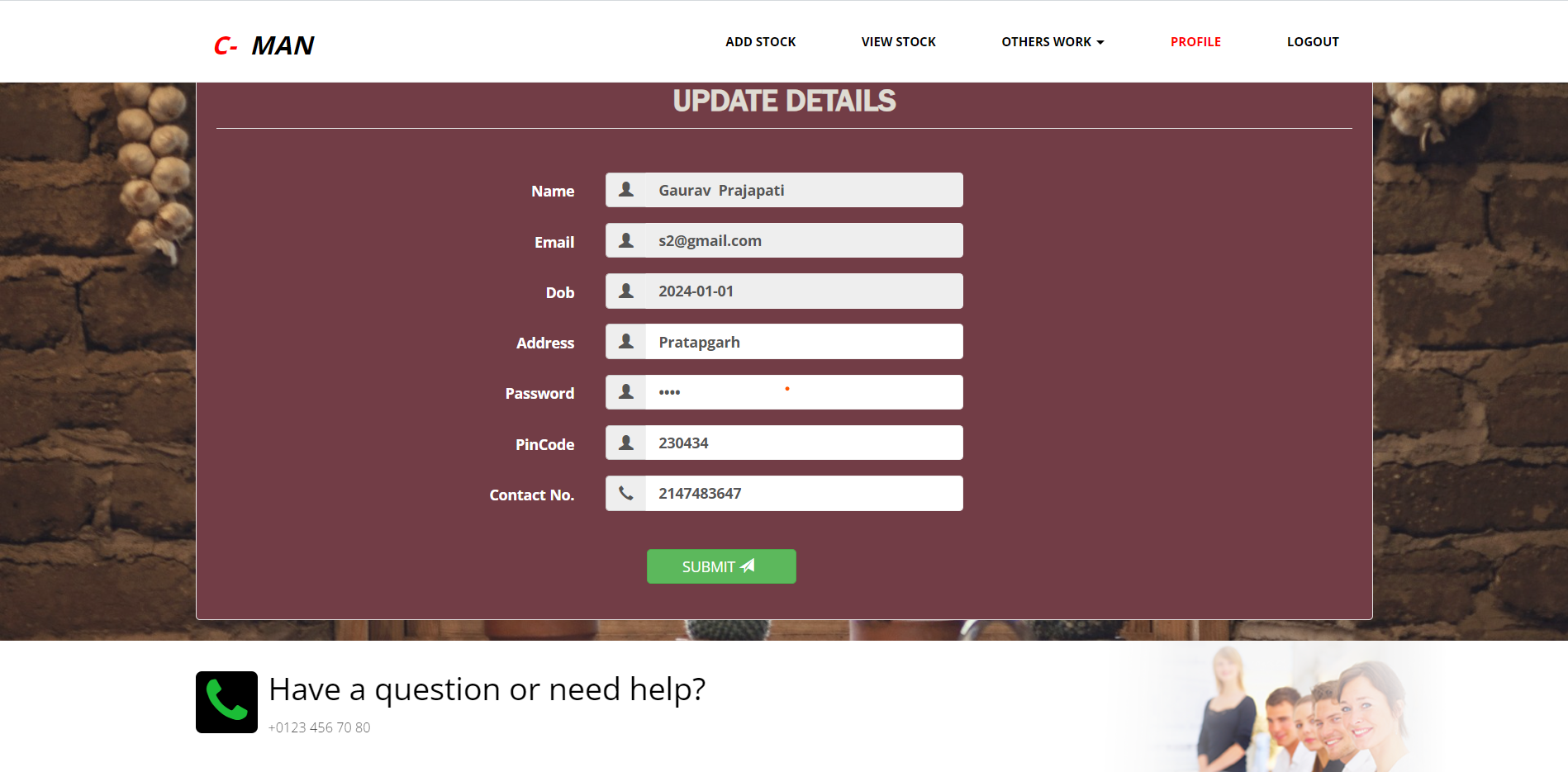
****

Fig 9.4 Profile Updation

Profile updation in the employee module of the Instagram Clone is a pivotal feature that users to maintain accurate and current personal information within the system. This functionality is accessible through the users module, where users can seamlessly navigate to the "Profile" section. Here, users have the ability to edit various aspects of their profiles, including personal details such as names, contact information, and emergency contacts. The system also facilitates password updates, enhancing security through periodic changes and adherence to password complexity standards. For Instagram with multiple shifts, users can manage their shift preferences and availability, allowing for a flexible scheduling approach. The profile up dation process includes a confirmation and review step, ensuring that users can verify the accuracy of the modifications made before finalizing the changes. An essential aspect of this feature is the creation of an audit trail, capturing the date, time, and specific changes made by the users.

**CHAPTER 10**

**TESTING**

**10.1 Introduction**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionalities of components, sub-assemblies, and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**10.2 Types of Testing**

**10.2.1 Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

**10.2.2 Integration Testing**

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

**10.2.3 System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT**

* Further enhancements to this project will be to Message Alert on Stock availability is less.
* Another enhancement would be allowed for users also.
* Add a feature for users to provide feedback and ratings for the profiles and service. Use this feedback to improve the quality of service and address any issues.
* Implement advanced analytics and reporting features to track news feed, popular menu feeds, peak hours, and other key metrics. Use this data to make informed business decisions and optimize operations.

**CONCLUSION & REFERNCES**

We have successfully designed a project on Web Based Instagram Clone. This project is designed in Django.

Throughout the development process, we encountered various challenges, including learning curve associated with Django framework, intricacies of front-end design with HTML and CSS, and managing database interactions. However, through diligent research, effective communication, and teamwork, we were able to overcome these obstacles and deliver a fully functional prototype.

Overall, the Instagram clone project has been a rewarding journey that has enriched our understanding of web development concepts and equipped us with practical skills that will undoubtedly be beneficial in our academic and professional endeavors.

Keeping in view these facts we will develop successfully. Developing the project will help us some experience on website development.   
**References**

Coding phase: **-**

1. Django (Complete Reference)

Referenced Sites:

* <https://www.w3schools.com/html/>
* <https://docs.djangoproject.com/en/>