LinkedNet

Project Report Submitted by

A BILAHARI

Reg. No.: AJC21MCA-2003

In Partial fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS

(MCA TWO YEAR)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with ‘A’ grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2022-2023

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**LINKEDNET**" is the bonafide work of **A BILAHARI (Regno: AJC21MCA-2003)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2022-23.

Mr. Rony Tom
Internal Guide

Ms. Meera Rose Mathew
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

DECLARATION

I hereby declare that the project report **“LINKEDNET”** is a bonafide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2022-2023.

Date:	**A Bilahari**
Kanjirappally	**Reg: AJC21MCA-2003**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude and appreciation to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude and appreciation towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for the overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Mr. Rony Tom** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

A Bilahari

ABSTRACT

The goal of this project is to implement a web application, which represents a company social networking application, similar to LinkedIn. In the application there are 3 roles: Administrator, Company and user. Admin has the right to manage the whole website. Companies manage information about their education and work experience, make requests to connect with other companies and accept or reject connection requests, review and manage the network of affiliated companies, post articles that can contain images and videos, post and respond to job recruitment posts, see in their timeline the articles posted by affiliate companies having the opportunity to comment on them, note their interest in specific posts and be notified of other users' interest in their posts, have private discussions with their affiliates companies, navigate the presentation pages of other companies, manage their login settings. Users can post photos, like photos posted by other users, chat with other users, comment on the posts and apply for jobs.

Some of the core functionalities offered are:

- User Registration

  This system provides users to register their various types of profiles like social, personal, general, company.

- Profile Management

  This system provides users to upload the photos. This system provides users to maintain their friend list. This system provides companies to post job vacancies.

- Admin Functionality

  Admin has the whole power to manage the system. He has the power to block users or company, delete posts, delete likes and comments

CONTENT

List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language.
CSS	-	Cascading Style Sheet
SQL	-	Structured Query Language
UML	-	Unified Modeling Language
RDBMS	-	Relational Database Management System

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The goal of this project is to implement a web application, which represents a company social networking application, similar to LinkedIn. In the application there are 3 roles: Administrator, Company and user. Admin has the right to manage the whole website. Companies manage information about their education and work experience, make requests to connect with other companies and accept or reject connection requests, review and manage the network of affiliated companies, post articles that can contain images and videos, post and respond to job recruitment posts.

1.2 PROJECT SPECIFICATION

This project aims to develop a web application that simulates a LinkedIn-like social networking service. It is a sizable project that includes numerous functions to improve the general look and feel of the website, including friend list organising, profile updating, and various other applications. There are three roles in this application: administrator, company, and user. Users can upload images and create various types of profiles using this system, including social, personal, and business profiles. This system enables employers to post open positions. Users can search for jobs by firm name or title.

Users

- Admin
- Company
- User

Functionalities

- User Registration

  Users can register and login, after logging in they can view photos posted by other users and apply for jobs posted by company.
- Profile Management

  Profile management module maintain the profile of a user like name, like, dislike, hobbies, status, etc.
- Friends Organization

  This module maintains the friends list, handles request and sends request to the other user.
- Admin Functionality

  Admin has the whole power to manage the system. He has the power to block users, delete post, delete likes.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

This project aims to develop a web application that simulates a LinkedIn-like social networking service. It is a sizable project that includes numerous functions to improve the general look and feel of the website, including friend list organising, profile updating, and various other applications. There are three roles in this application: administrator, company, and user. Users can upload images and create various types of profiles using this system, including social, personal, and business profiles. This system enables employers to post open positions. Users can search for jobs by firm name or title.

2.2 EXISTING SYSTEM

2.2.1 NATURAL SYSTEM STUDIED

The natural system is for open positions to be advertised as such in publications and other posters. To inquire about job availability, there are several advertising firms available. Since every process in the existing system is performed by a human, it naturally takes more time.

2.2.2 DESIGNED SYSTEM STUDIED

The designed system is a comprehensive information system made up of numerous electronic technical techniques that support social and commercial operations. A user can register, login to the website, and conduct a search for individuals or businesses using the system as it is intended. After then, people can submit applications for positions offered by companies.

2.3 DRAWBACKS OF EXISTING SYSTEM

- The existing system was not very effective & was highly time consuming.
- It is difficult to find job listings for the company we are looking for.
- Lack of accuracy

2.4 PROPOSED SYSTEM

The proposed system is designed to address every drawback of the current system. Our plan will include a website-like online interface that will let users communicate with one another and apply for jobs listed by businesses. Users of this system can look up and follow other users or businesses. LinkedNet was created to help businesses expand more quickly. The user-friendly design makes it easy for users to accomplish their goals. In that situation, all paper work will be eliminated by switching manual work to digital work. The administrator can easily maintain the security and accuracy. Companies may advertise employment openings. Users have the ability to publish images, like and comment on other users' posts, and apply for positions that businesses post.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- Efficient.
- Ensure data accuracy.
- Save a lot of time and effort.
- User friendly system.
- Security of data.
- Flexible to use the system.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study is a detailed analysis that considers all of the critical aspects of a proposed project in order to determine the likelihood of its success. A feasibility study is an assessment of the practicality of a proposed plan or project. A feasibility study analyzes the viability of a project to determine whether the project or venture is likely to succeed. The study is also designed to identify potential issues and problems that could arise while pursuing the project.

The document provides the feasibility of the proposed system that is being developed and three aspects of the feasibility of a project are considered such as technical, economical, and behavioral during the feasibility study.

3.1.1 Economic Feasibility

Economic feasibility analysis is the most commonly used method for determining the efficiency of a new project. It is also known as cost analysis. It helps in identifying profit against investment expected from a project. Cost and time are the most essential factors involved in this field of study.

The economic feasibility study for projects shows all the costs necessary for the project, which are the costs of establishing the project, as well as the size of the investment, estimating the size of the project's profits, and knowing the net profit during a specific period of time. and tools needed for the project.

The proposed system is developed as part of project work, thus there is no manual cost spent for the proposed system. All the required resources were already available, which shows that the system is economically feasible for development. The project was developed at a low cost as it is completely developed using open-source software.

3.1.2 Technical Feasibility

The technical feasibility study can be defined as the study related to all the technical aspects of the project. It includes defining the technical specifications of the product and the size of the project, and preparing the necessary labor schedules for production.

In technical feasibility the following issues are taken into consideration.

- Whether the required technology is available or not
- Whether the required resources are available

3.1.3 Behavioral Feasibility

Behavioral feasibility answers the following questions:

- Whether the proposed system will cause any harm to its users?
- It is proposed system sufficient to support the users?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

3.1.4 Feasibility study questionnaire

1. Do you find the jobs you're looking for easily in newspaper ads?
    - No
2. How do you ensure that a company is genuine?
    - When a company is registered the admin verifies the details of the company. Only after the verification the account is created.
3. How is LinkedIn useful for students?
    - Students can apply for their dream jobs through this platform.
4. What are the benefits of LinkedIn Marketing?
    - Benefits are, companies can easily get employees and job seekers can easily apply for jobs in their dream company.
5. How LinkedIn is important for job seekers?
    - This easily increases the chance of being hired.
6. Can I grow my business on LinkedIn?
    - You can use it to build your brand's reputation and increase engagement with potential clients while also ensuring that you are driving enough traffic to your official website.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	- Intel Core i5
RAM	- 4 GB
Hard disk	- 1 T B

3.2.2 Software Specification

Front End	-	HTML, CSS
Backend	-	Django, SQLite
Client on PC	-	Windows 7 and above.
Technologies used	-	JS, HTML5, Django, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

Django is an open-source web framework written in the Python programming language. Named after the jazz guitarist Django Reinhardt, it is used by some of the largest websites in the world including Instagram, Mozilla, and NASA, but also lightweight enough to be a popular choice for weekend side projects and startups. Its "batteries-included" approach means a powerful website can be generated quickly in the hands of a skilled developer

Django adopts a "batteries-included" approach similar to Python and comes with a number of built-in features including an extensible authentication system, robust admin app, lightweight testing web server, and support for multiple databases including PostgreSQL, MySQL, MariaDB, Oracle, and SQLite. It is known for its leading security best practices and comes with comprehensive documentation, available either online or as a PDF/ePUB for offline consumption. As a mature project, Django rarely makes breaking changes and has a clear deprecation schedule for any updates. A major new version is released every nine months or so with monthly patch releases for security and bug fixes. There is also a vibrant ecosystem of third-party applications visible on the Django Packages site--which provide additional functionality. Over time, the most popular packages are often rolled into Django itself.

3.3.2 SQLite

SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine in the world. It is an in-process library and its code is publicly available. It is free for use for any purpose, commercial or private. It is basically an embedded SQL database engine. Ordinary disk files can be easily read and write by SQLite because it does not have any separate server like SQL. The SQLite database file format is cross-platform so that anyone can easily copy a database between 32-bit and 64-bit systems. Due to all these features, it is a popular choice as an Application File Format.

Applications of SQLite

- Due to its small code print and efficient usage of memory, it is the popular choice for the database engine in cellphones, PDAs, MP3 players, set-top boxes, and other electronic gadgets.
- It is used as an alternative for open to writing XML, JSON, CSV or some proprietary format into disk files used by the application.
- As it has no complication for configuration and easily stores file in an ordinary disk file, so it can be used as a database for small to medium sized websites.
- It is faster and accessible through a wide variety of third-party tools, so it has great application in different software platforms.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

System Design is the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. system design ranges from discussing the system requirements to product development. System development creates or alters the system so that the processes, practices, and methodologies are changed to develop the system. Therefore, a systematic approach is needed to manage the system requirements and design methodology. It can be classified as logical design and physical design. The logical design represents the abstract dataflow, while the physical design represents the system's input and output processes. It specializes in developing great artwork by saving time and effort. This helps in creating plans for information systems. It is used to solve internal problems, boost efficiency, and broadcast opportunities. It also is the foundation of any business. It contributes a lot to successfully achieving the required results and makes working easier and simpler.

4.2 UML DIAGRAM

The components of the principles of object-oriented programming are represented by the language known as the Unified Modeling Language (UML), which is utilized in the industry of software engineering. It serves as the standard definition of the entire software architecture or structure. Complex algorithms are solved and interacted with in Object-Oriented Programming by treating them as objects or entities. Anything can be one of these things. It could either be a bank manager or the bank itself. The thing can be a machine, an animal, a vehicle, etc. The issue is how we connect with and control them, even though they are capable of and ought to execute duties. Interacting with other objects, sending data from one object to another, manipulating other objects, etc., are examples of tasks. There could be hundreds or even thousands of objectsin a single piece of software

UML diagram includes the following diagrams:

- Use case Diagram
- Class diagram
- Object Diagram
- Sequence Diagram
- Activity Diagram
- Statechart Diagram
- Deployment Diagram
- Component

4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between system components. An approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" in this context refers to a project or business that is under development or operation, such a mail-order goods sales and service web page. The Unified Modelling Language (UML) makes use of use case diagrams. a common notation for simulating systems and things in the actual world. Planning for overall requirements is one of the system objectives. Testing and debugging a software product, and verifying a hardware design Performing a consumer service, developing, writing an online help guide, or focused task Use cases in a product sales context, for instance, would include ordering of goods, catalogue revision, transaction processing, and client.

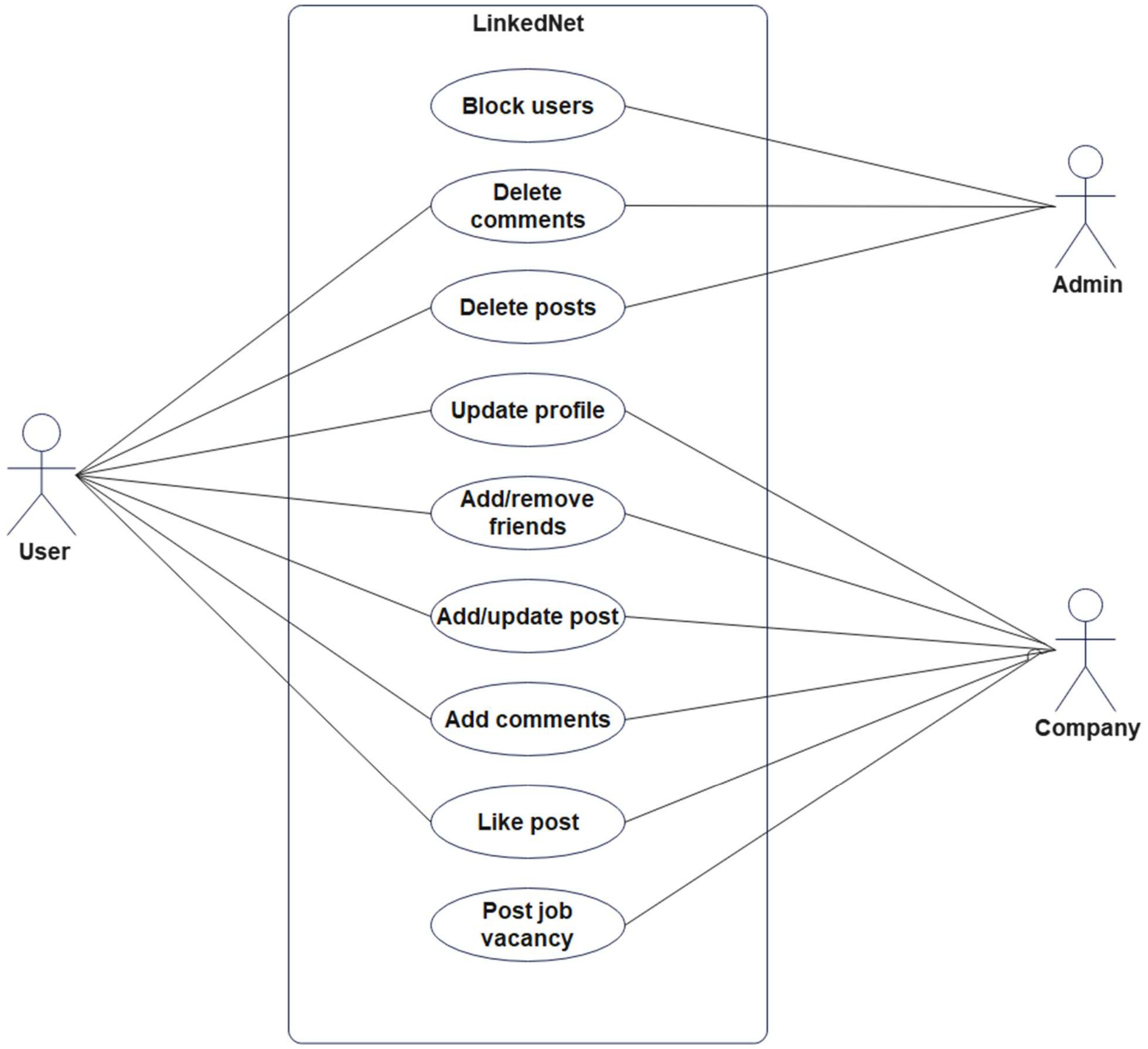


Figure:4.2.1 Use case Diagram

4.2.1 SEQUENCE DIAGRAM

A sequence diagram essentially shows how things interact with one another sequentially, or the order in which these interactions occur. A sequence diagram can also be referred to as event diagrams or event scenarios. Sequence maps define the actions that the system's components take and in what order. These schematics are Businesspeople and software developers frequently employ documentation and understanding specifications for both current and future systems.

Sequence Diagram Notations –

Actors – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

Lifelines – A lifeline is a named element which depicts an individual participant in asequence diagram. So basically, each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram

Messages – Communication between objects is depicted using messages The messagesappear in a sequential order on the lifeline. We represent messages using arrows. Lifelinesand messages form the core of a sequence diagram.

Guards – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

User

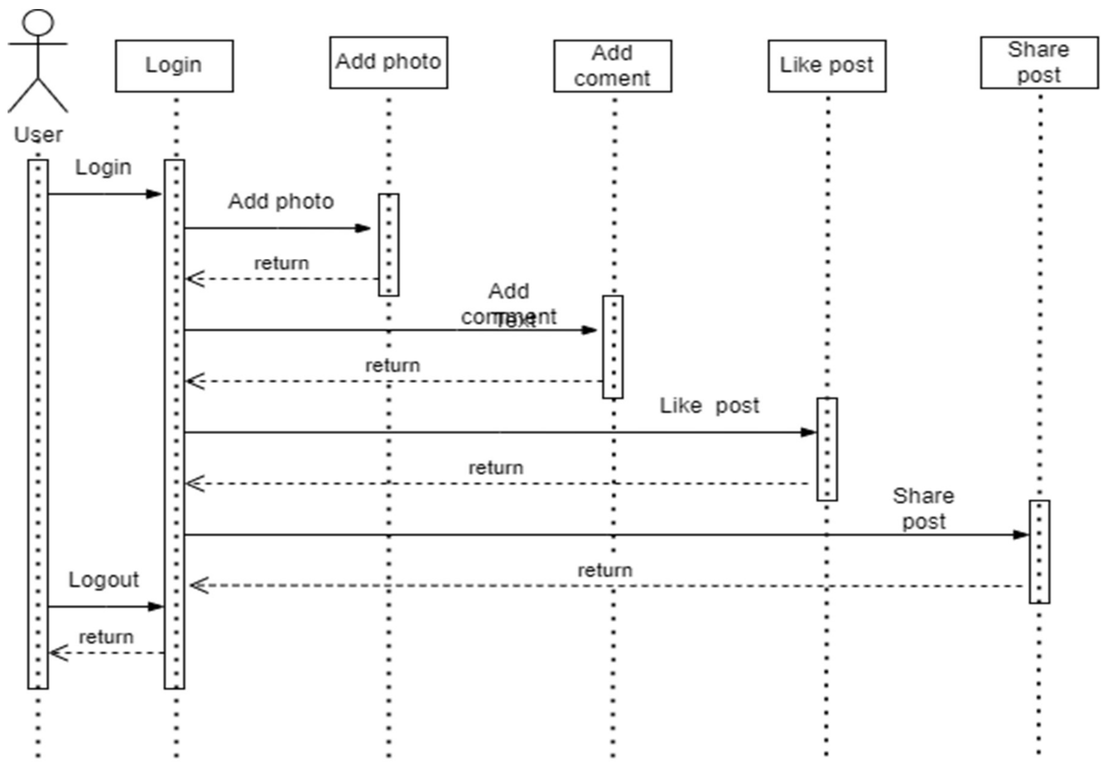


Figure:4.2.2 Sequence Diagram

Admin

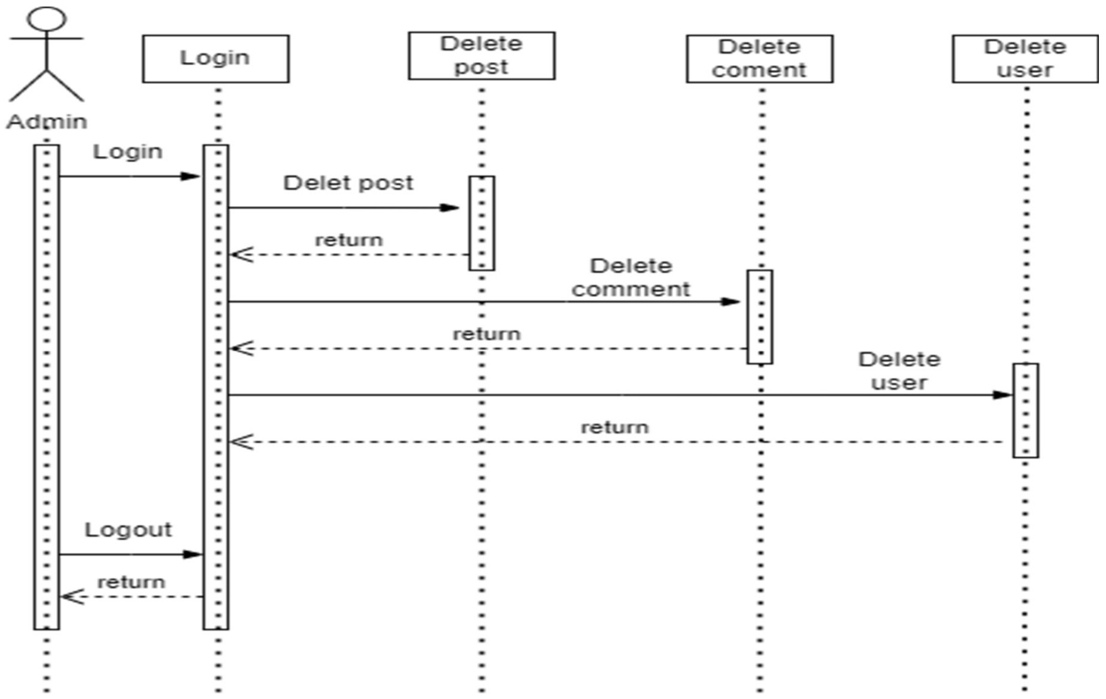


Figure:4.2.2 Sequence Diagram

Company

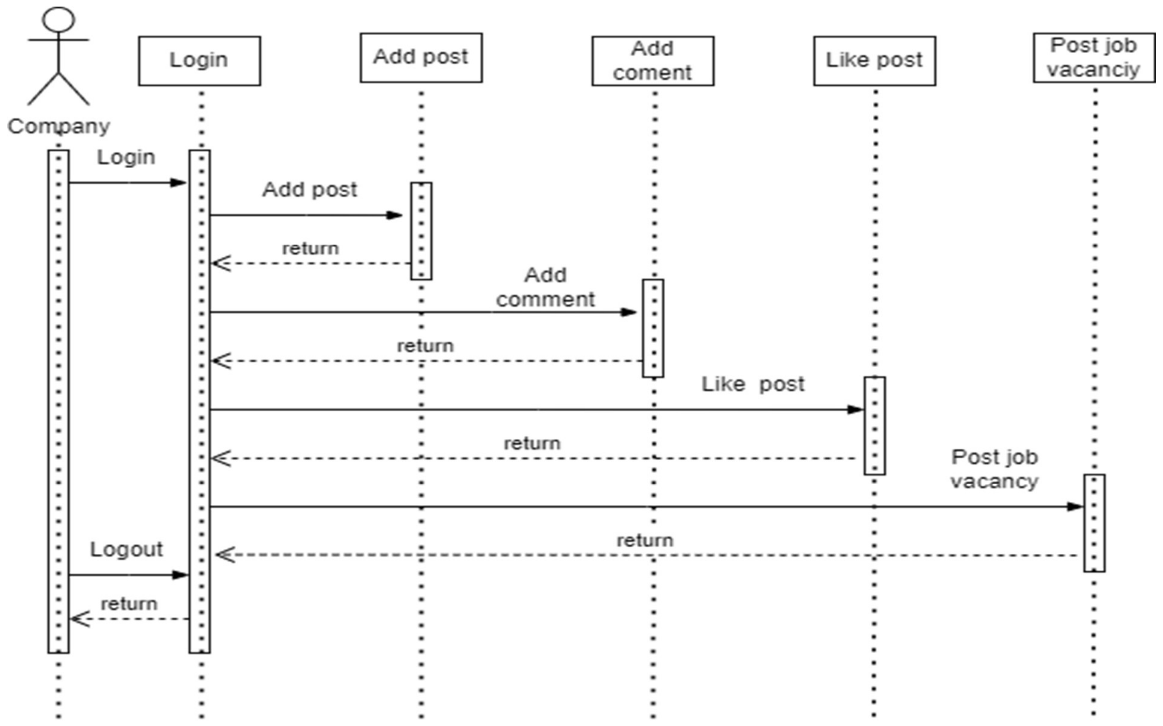


Figure:4.2.2 Sequence Diagram

4.2.2 State Chart Diagram

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics. The Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events. Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of the Statechart diagram is to model the lifetime of an object from creation to termination. Statechart diagrams are also used for the forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

The main purposes of using Statechart diagrams are:

- To model the dynamic aspect of a system.
- To model the lifetime of a reactive system.
- To describe different states of an object during its lifetime.
- Define a state machine to model the states of an object.

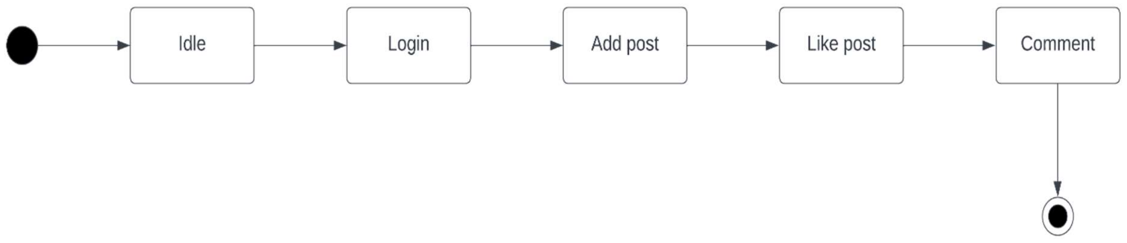


Figure:4.2.3 State Chart Diagram

4.2.3 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores. Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types are:

- Ellipses represent actions
- Diamonds represent decisions.
- Bars represent the start (split) or end (join) of concurrent activities.
- Black circle represents the start (initial node) of the workflow.
- Encircled black circle represents the end (final node).

Arrows run from the start towards the end and represent the order in which activities happen. Activity diagrams can be regarded as a form of a structured flowchart combined with a traditional data flow diagram. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

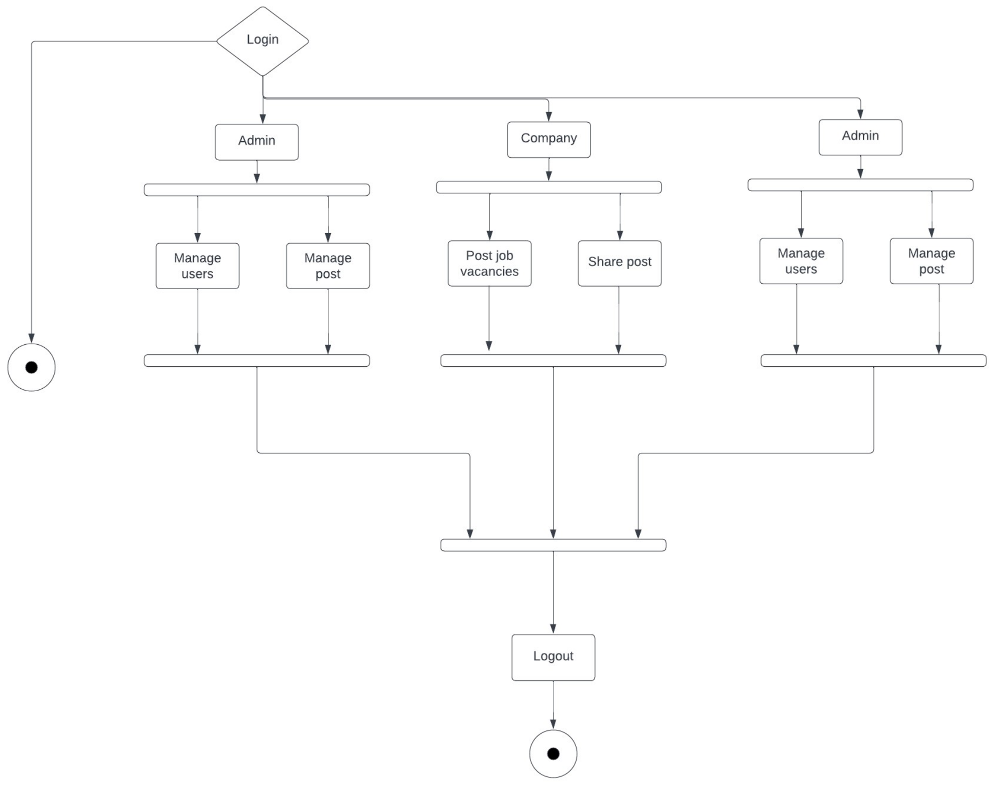


Figure:4.2.4 Activity Diagram

4.2.5 Class Diagram

Class diagrams is a Static diagram. It represents the application's static view. Class diagrams are used to create executable code for software applications as well as for visualizing, explaining, and documenting various elements of systems. The characteristics and functions of a class are described in a class diagram, along with the restrictions placed on the system. Because they are the only UML diagrams that can be directly transferred to object-oriented languages, class diagrams are frequently employed in the modelling of object-oriented systems. A collection of classes, interfaces, affiliations, collaborations, and constraints are displayed in a class diagram.

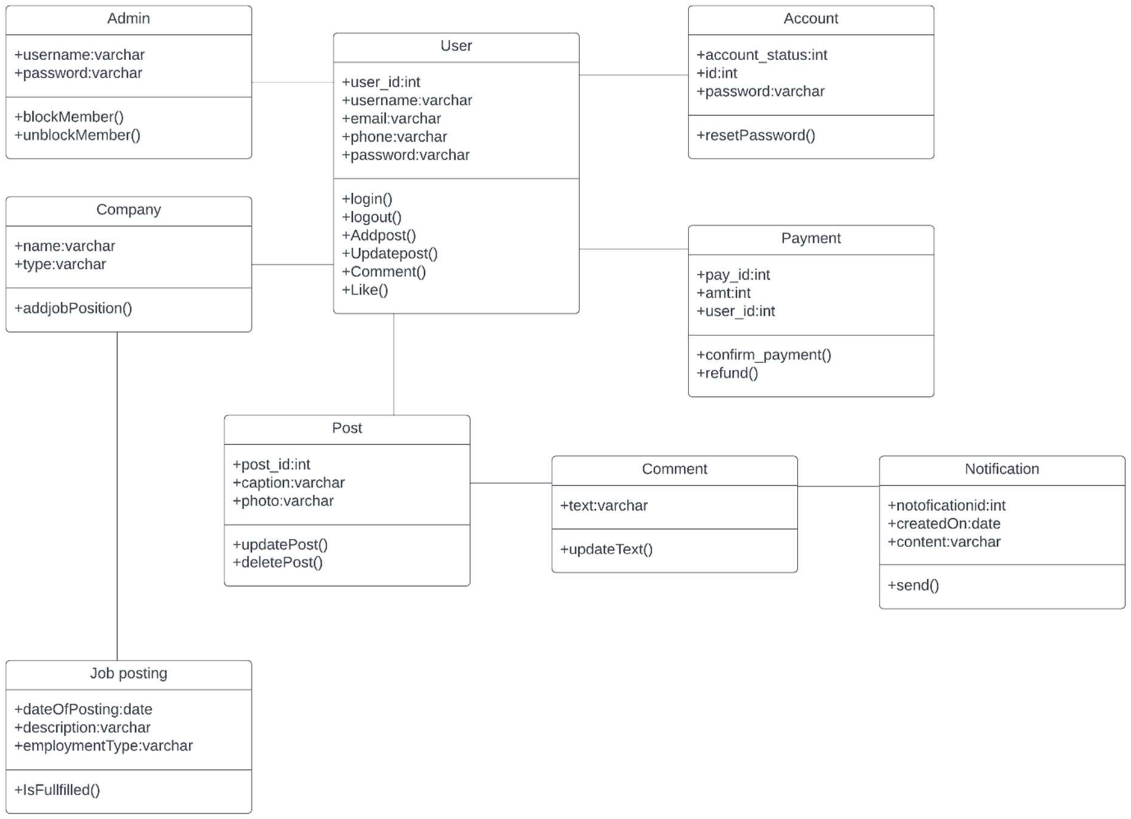


Figure:4.2.5 Class Diagram

4.2.6 Object Diagram

Class diagrams are a requirement for object diagrams because they are the source of class diagrams. An object diagram illustrates a specific instance of a class diagram. The basic concepts of class diagrams and object diagrams are the same. Object diagrams are also used to describe a system's static view, which is a snapshot of the system taken at a particular point in time. You can see a group of things and their relationships by using object diagrams.

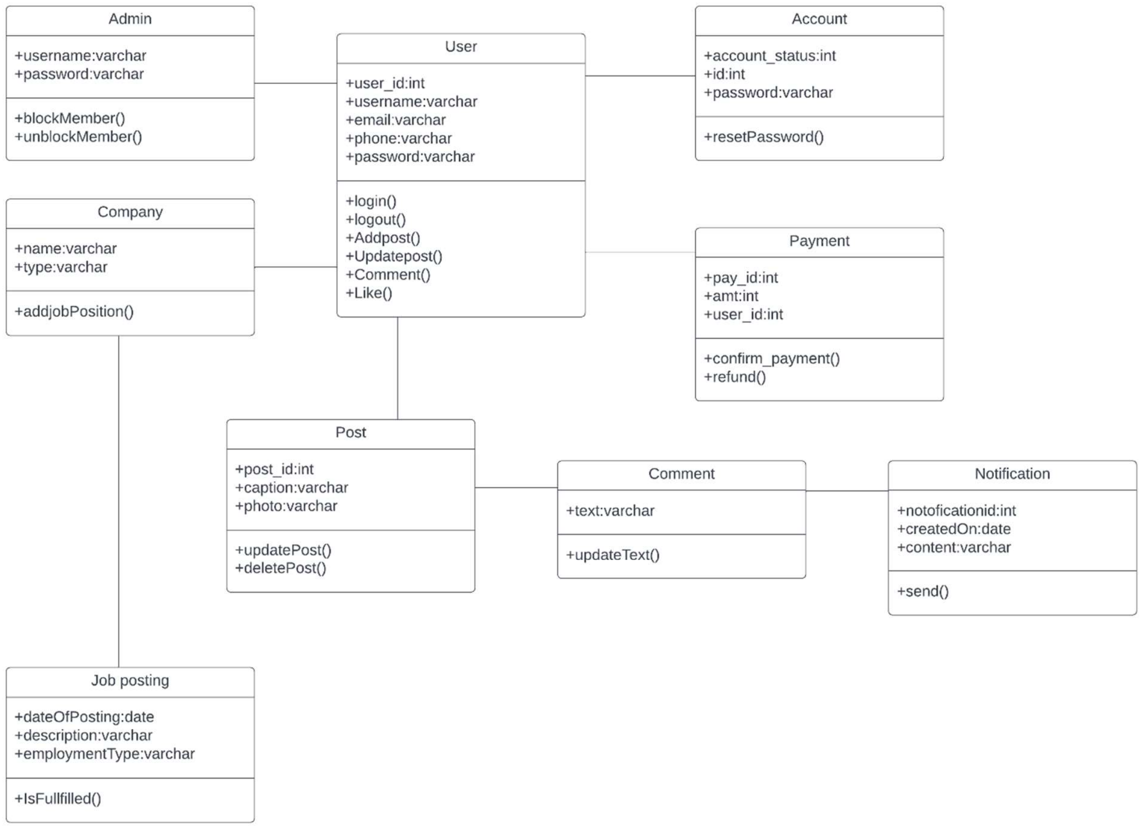


Figure:4.2.5 Object Diagram

4.2.7 Component Diagram

Component diagrams come in a variety of behaviors and personalities. The physical parts of the system are represented using component diagrams. Executables, libraries, files, documents, and other items that are physically present in a node are just a few examples. Component diagrams are used to show how the components of a system are connected and arranged. These diagrams can also be used to construct systems that can be run.

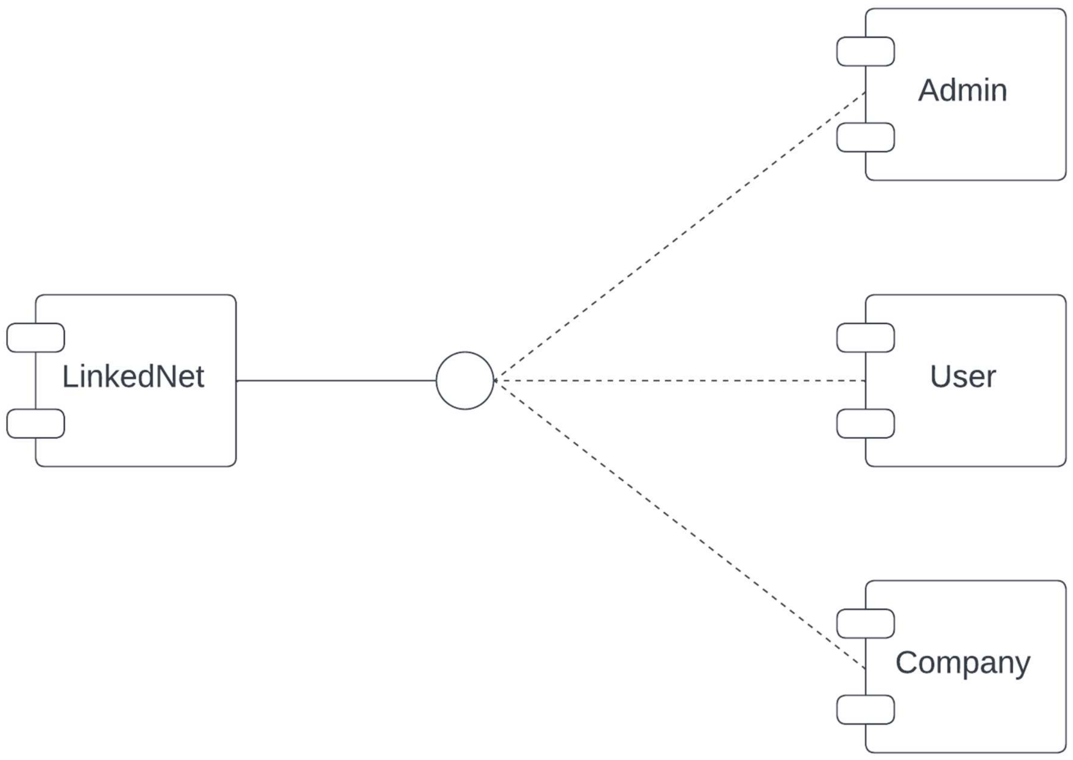


Figure:4.2.7 Component Diagram

4.2.8 Deployment Diagram

Deployment diagrams show the topology of a system's physical components, where the software components are installed. Deployment diagrams are used to describe a system's static deployment view. The key elements of deployment diagrams are nodes and connections between them.

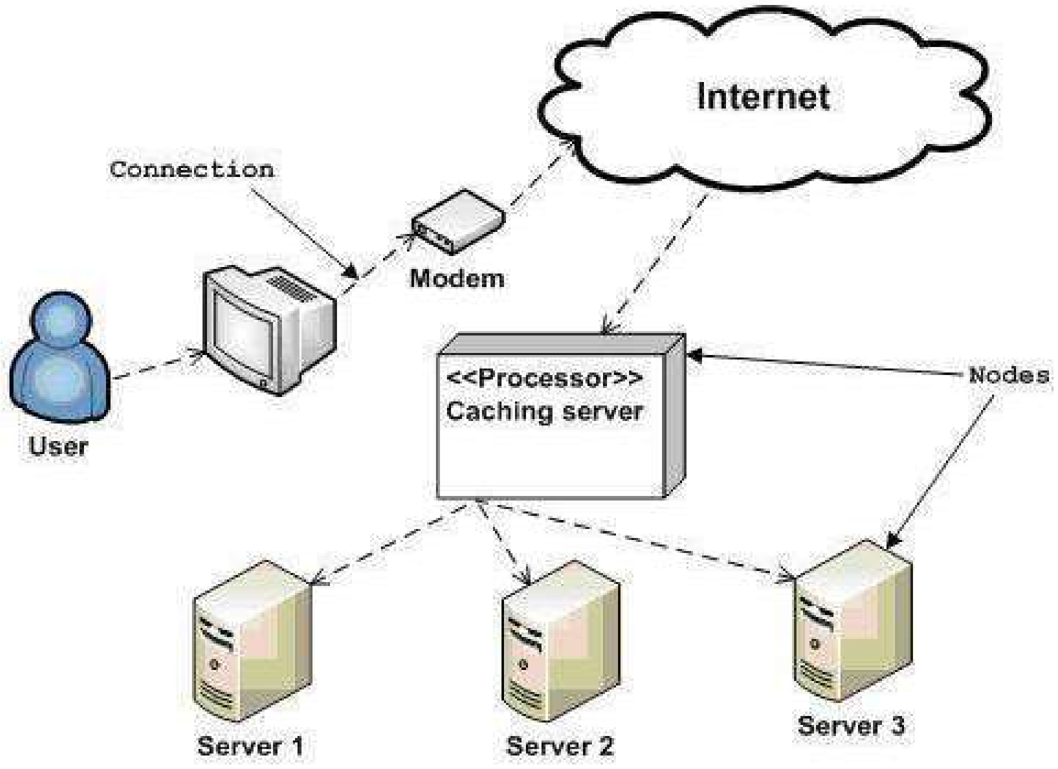


Figure:4.2.8 Deployment Diagram

4.3 USER INTERFACE DESIGN USING FIGMA

4.3.1 Form Name: User Login

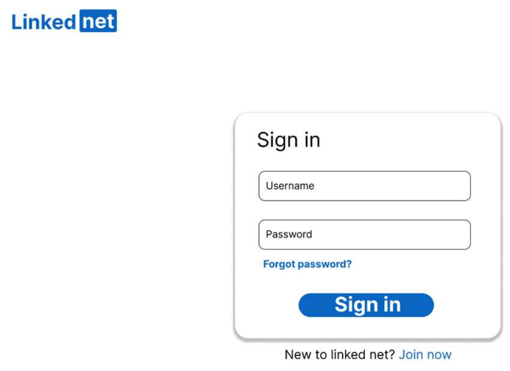


Figure:4.3.1

4.3.2 Form Name: User Registration

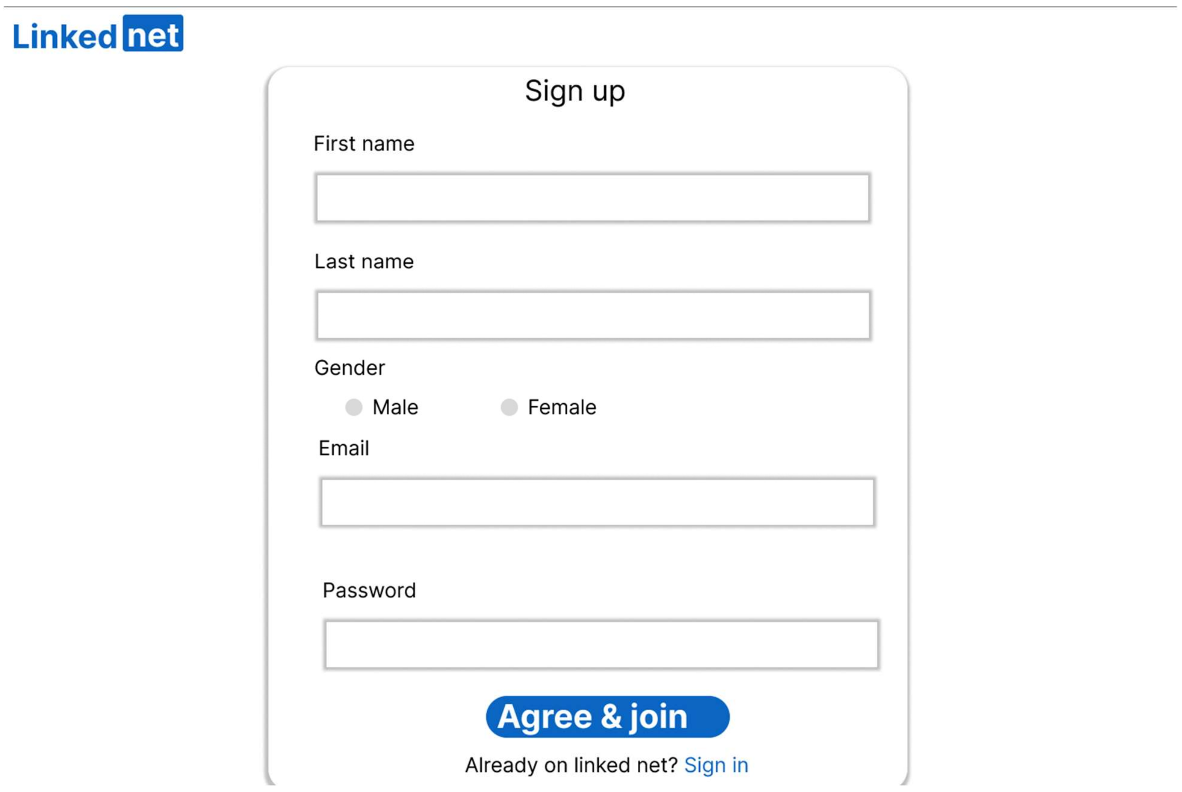


Figure:4.3.2

4.3.3 Form Name: User Home Page

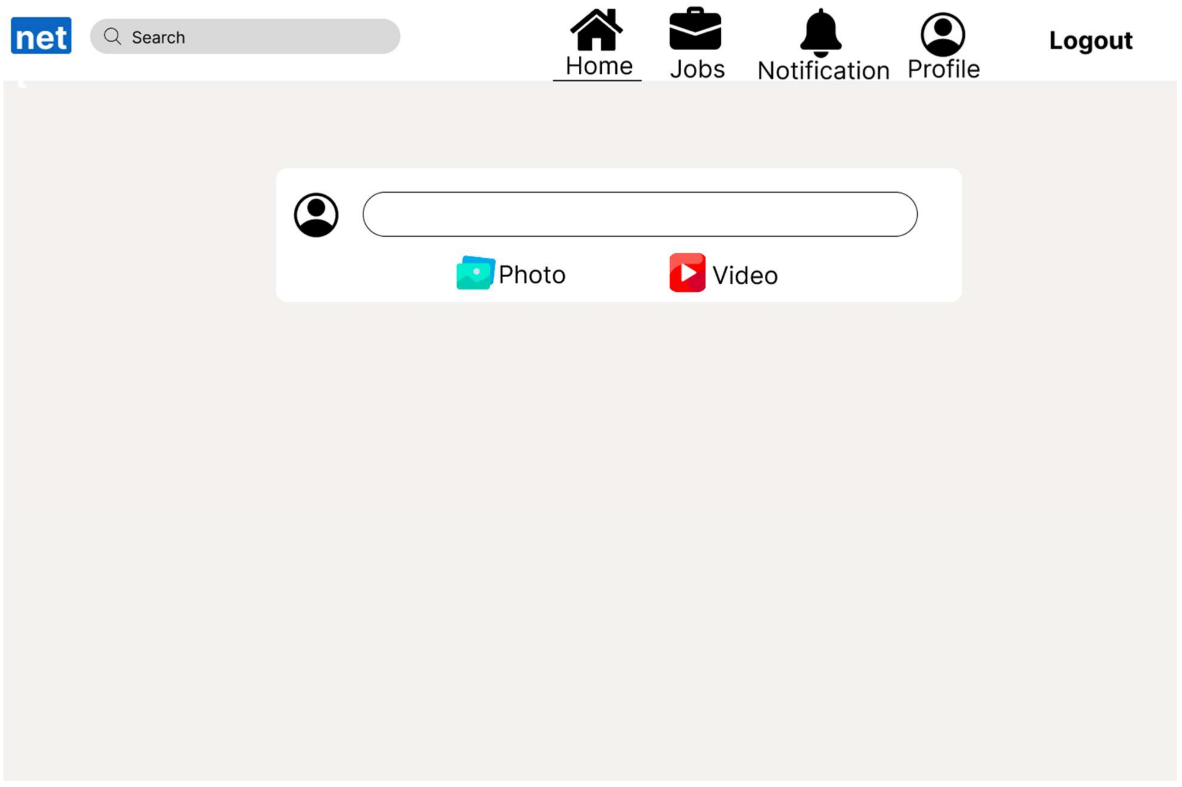


Figure:4.3.3

4.4 DATABASE DESIGN

A database is a structured system with the ability to store information and allow users to access that information quickly and effectively. Any database must be protected because its primary goal is its data. The database design process has two stages. User requirements are obtained in the first step, and a database is created to as clearly as possible meet these objectives. Information Level Design is the name of this stage, which is carried out independently of any specific DBMS. The second phase involves converting this information level design into a design for the DBMS that will be used to implement the system in issue. Physical Level Design is the stage where the properties of the particular DBMS are considered. Data Integrity Data independence

4.4.1 Relational Database Management System (RDBMS)

In a relational model, the database is shown as a set of relations. each connection like a file or table of records with values. formal terminology for a relational model, A column header is known as an attribute, a row is known as a tuple, and the table is known as a relation.

Each table in a relational database is made up of data that is stored in rows and columns. assigned an arbitrary name. In a story, each row represents a group of associated values.

Domains, Relations, and Attributes

A relation is a table. Tuples are the units of a table's rows. An ordered group of n elements is a tuple. Attributes are referred to as columns. Every table in the database has relationships already established between them. This guarantees the integrity of both referential and entity relationships. A group of atomic values make up a domain D. Specifying the data type from which the domain's data values are derived is a standard way to define a domain. To make it easier to understand the values of the domain, it is also helpful to give it a name. Each value ina relation is atomic and cannot be broken down.

4.4.2 Normalization

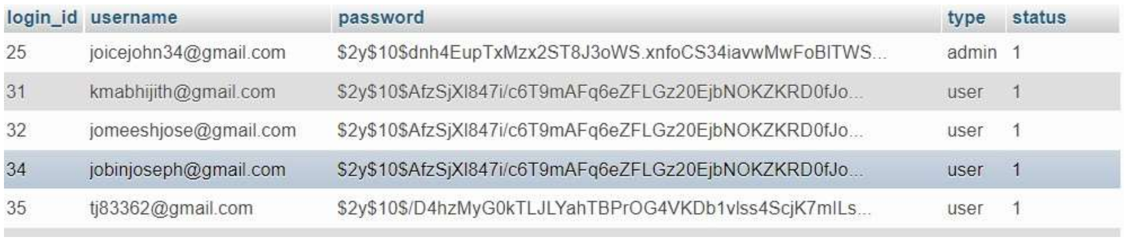
The simplest possible grouping of data is used to put them together so that future changes canbe made with little influence on the data structures. Data normalization is a formal process. structures in ways that encourage integrity and remove duplication. Normalization is a method of dividing large datasets into smaller ones and removing superfluous fields. Into a smaller table. Additionally, it serves to prevent additions, deletions, and updates. Anomalies. Keys and relationships are two notions commonly used in data modelling. A table row is uniquely identified by its key. key uniquely identifies a row in a table. A primary key is an element, or set of components, in a table that serves as a means of distinguishing between records from the same table. A column in a table known as a foreign key is used to uniquely identify records from other. tables. Up to the third normal form, all tables have been normalized means placing things in their natural form, as the name suggests. By using normalization, the application developer aims to establish a coherent arrangement of the data into appropriate tables and columns, where names may be quickly related to the data by the user.

First Normal Form

According to the First Normal Form, any attribute's tuple's value must be a single value from itsdomain, which must only contain atomic values. the territory of that property. To put it another way, 1NF forbids "relations within relations. "Alternatively, "relations as attribute values within tuples." The sole attribute values that are allowed by1NF are indivisible or single-atom values. The data must be entered into Initial as the first step. Standard Form.
By putting the data in separate tables, you may donate this.in each table is of a similar type. A primary key or foreign key is assigned to each table as per the project's requirements. For each non-atomic relationship, we create new ones in this. attribute or nested relation. This eliminated

repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only. connection or nested attribute. This got rid of data groups that were repeated.

Example: tbl_login



login_id	username	password	type	status
25	joicejohn34@gmail.com	$2y$10$dnh4EupTxMzx2ST8J3oWS.xnfoCS34iavwMwFoBITWS...	admin	1
31	kmabhijith@gmail.com	$2y$10$AfzSjXl847i/c6T9mAFq6eZFLGz20EjbNOKZKRD0fJo...	user	1
32	jomeeshjose@gmail.com	$2y$10$AfzSjXl847i/c6T9mAFq6eZFLGz20EjbNOKZKRD0fJo...	user	1
34	jobinjoseph@gmail.com	$2y$10$AfzSjXl847i/c6T9mAFq6eZFLGz20EjbNOKZKRD0fJo...	user	1
35	tj83362@gmail.com	$2y$10$/D4hzMyG0kTLJLYahTBPrOG4VKDb1vlss4ScjK7mILs...	user	1

Second Normal Form

Accordance with Second Normal Form No non-key attribute should be functionally dependent on a portion of the primary key for relations when the main key has several attributes. This involves breaking down each partial key into its dependent characteristics and setting up a new relation for each one. Keep the original primary key and any properties that are entirely dependent on it in your database. This procedure aids in removing data that depends only on a small portion of the key. If and only if a relation satisfies all the requirements for first normal form for the primary key and every non-primary key attribute of the connection is completely dependent on its primary key alone, then that relation is said to be in second normal form.

Third Normal Form

According to the Third Normal Form relation should not have a non-key attribute that is functionally determined by another non-key attribute or by a collection of non-key attributes. The primary key should not be transitively dependent, in other words. The non-key attributes that functionally determine other non-key attributes are decomposed in this way put up in relation. This action is made to remove anything that is not completely dependent on the Primary Key. Only when a relation is in second normal form and, more importantly, when its non-key characteristics do not depend on those of other non-key attributes, is it considered to be in third normal form.

4.4.3 Sanitization

An automated procedure called "sanitization" is used to get a value ready for use in a SQL query. This process typically involves checking the value for particular characters that have a special significance for the target database. To prevent a SQL injection attack, you must sanitize (filter) the input string while processing a SQL query based on user input. For instance, the userand password input are a typical scenario. In that particular scenario, the server response would provide access to the 'target user' account without requiring a password check.

4.4.4 Indexing

By reducing the number of disk accesses needed when a query is completed, indexing helps a database perform better. It is a data structure method used to locate and access data in a database rapidly. Several database columns are used to generate indexes. The primary key or candidate key of the table is duplicated in the first column, which is the Search key. To make it easier to find the related data, these values are kept in sorted order Recall that the information may or may not be kept in sorted order.

4.5 TABLE DESIGN

1.Table Name: **Tbl_user**

Use: To store the user login details

Primary key: **uid**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	uid	Int	Primary Key	Id of User
2	password	Varchar (128)		Password of User
3	last_login	datetime		Last login date and time
4	first_name	Varchar(90)		First name of user
5	last_name	Varchar(90)		Last name of user
6	username	Varchar(150)		Username of the user
7	is_superuser	bool		If user is admin

2. Table Name: **Tbl_company**

Use: To store the company details

Primary key: c**id**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	cid	Int	Primary Key	Id
2	cname	Varchar (100)		Company’s name
3	email	Varchar (100)		Company’s mail
4	password	Varchar (30)		Password of company
5	website	Varchar (200)		Website of company
6	username	Varchar(50)		Username of company
7	Location	Varchar(100)		Company location

3. Table Name: **Tbl_followerscount**

Primary key: f**id**

Use: To store the follower Details

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	fid	Int	Primary Key	Id
2	follower	Varchar (20)		Follower
3	user	Varchar (20)		User who is followed

4. Table Name: **Tbl_likepost**

Primary key: lid

Use: To store the likes of post

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	lid	Int	Primary Key	like Id
2	Post_id	Int	Foreign Key	Post id, foreign key
3	username	Varchar (100)		Username of the user who liked the post

5. Table Name: **Tbl_post**

Primary key: pid

Use: To store the post details

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	Post_id	Int	Primary Key	post Id
2	user	Varchar (100)		User who posted
3	image	Varchar (100)		Posted image
4	caption	Varchar (500)		Caption of the post

6. Table Name: **Tbl_profile**

Primary key: pro_id

Use: To store the user profile details

Foreign Key: uid tbl_user

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	Pro_id	Int	Primary Key	Profile id
2	uid	Int	Foreign Key	User id, Foreign key
3	bio	varchar (100)		Bio
4	Phone	Varchar(10)		Phone number
5	Profileimg	Varchar(300)		Profile image
6	Location	Varchar(100)		Location of the user
7	Tenth	Int		Tenth cgpa
8	Twelfth	Int		Twelfth cgpa
9	Ug	Int		Ug cgpa
10	Pg	Int		Pg cgpa

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing is the procedure of meticulously monitoring the way software is used to seeif it works as expected. Software testing is usually used in conjunction with the word's validation and verification. A product, including software, is validated by being examined or evaluated to see if it conforms to all pertinent requirements. Software testing, one sort of verification, also makes use of reviews, analyses, inspections, and walkthroughs. Validationis the process of ensuring that what has been specified corresponds to what the user actuallywants.

Other procedures that are typically connected to software testing include static analysis and dynamic analysis. Without actually running the code, static analysis examines the software's source code to look for errors and gather statistics. Dynamic analysis looks at the behavior of software while it is in use to provide information like execution traces, timing profiles, and test coverage specifics.

Running a programme with the intention of finding any faults is how a programme is tested. Testing is a group of tasks that can be organized in advance and completed in a systematic way. Individual modules are tested first, followed by the integration of the entire computer-based system. There are numerous regulations that can be utilized as testing objectives, and testing is essential for the achievement of system testing objectives. The following:

- A test case with a high likelihood of detecting an unknown fault qualifies as a goodtest case.
- A test that finds an error that has not yet been found is successful.

A test that effectively accomplishes the aforementioned goals will reveal software problems. Testing also reveals that the software functions appear to function in accordance with the specification and that the performance requirements appear to have been met. There are three ways to test programmes.

- For accuracy
- For effective implementation
- Regarding computational complexity

Testing for correctness is meant to ensure that a program performs exactly as it was intendedto. This is much harder than it might initially seem, especially for big programs.

5.2 TEST PLAN

A test plan suggests a number of required steps that need be taken in order to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer program, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfils the purpose for which it was intended. In order to solve the inherent issues with allowing the builder to evaluate what they have developed, there is an independent test group (ITG). Testing's precise goals should be laid forth in quantifiable language. so that the cost to discover and remedy the problem, the mean time to failure. the cost to find and fix the defects, remaining defect density or frequency of occurrence and test workhours per regression test all should be stated within the test plan.

The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing
- Output Testing

5.2.1 Unit Testing

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as aguide when testing crucial control paths to find faults inside the module's perimeter. the level of test complexity and the untested area determined for unit testing. Unit testing is white- box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. To confirm that each statement in a module has been executed at least once, boundary conditions are evaluated. Finally, each path for managing errors is examined.

Testing of data flow through a module interface are important before beginning anyadditional tests. If data cannot correctly enter and exit the system, all other tests are useless.The unit test's selective analysis of execution routes is a crucial task. In order to cleanly reroute or stop work when an error does occur, error handling channels must be set up and error scenarios must be anticipated in excellent design. Boundary testing is the last stage ofunit testing.

In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome.

5.2.2 Integration Testing

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a program structure that has been determined by design using unit tested components. The program as a whole is tested. Correction is challenging since the size of the overall program makes it challenging to isolate the causes. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All of the modules were integrated after unittesting was completed in the system to check for any interface inconsistencies. A distinctiveprogram structure also developed when discrepancies in program structures were eliminated.

5.2.3 Validation Testing or System Testing

This marks the conclusion of the testing procedure. This required comprehensive testing ofthe system, which covered all forms, codes, modules, and class modules. System tests and black box testing are two common names for this kind of testing. The functional requirements of the software are the main emphasis of the black box testingapproach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every program requirement. erroneous or missing functions, interface faults, data structure or external data access errors,performance errors, initialization errors, and termination errors are the kind of issues that black box testing focuses on.

5.2.4 Output Testing or User Acceptance Testing

The system under consideration is tested for user acceptance; in this case, it must satisfy thebusiness' requirements. The programmer should consult the user and the perspective system as it is being developed in order to make any necessary adjustments. This is carried out in relation to the following things:

- Screen Designs for Input
- Screen Designs for Output

The aforementioned testing is carried out using a variety of test data. The preparation of test data is essential to the system testing process. The system under investigation is then put tothe test using the prepared test data. Errors in the system are once again found during testing,fixed using the methods described above, and logged for use in the future.

5.2.5 Automation Testing

Software and other computer goods are tested automatically to make sure they abide by tightguidelines. In essence, it's a test to ensure that the hardware or software performs exactly asintended. It checks for errors, flaws, and any other problems that might occur throughout the creation of the product. Any time of day can be used to do automation testing. It looks at the software using scripted sequences. It then summarizes what was discovered, and this data can be compared to results from earlier test runs.

Benefits of Automation Testing

Detailed reporting capabilities - Automation testing uses well-crafted test cases for various scenarios. These scripted sequences can be incredibly in-depth, and provide detailed reportsthat simply wouldn't be possible when done by a human.

Improved bug detection - One of the main reasons to test a product is to detect bugs and other defects. Automation testing makes this process an easier one. It's also able to analyzewider test coverage than humans may be able to.

Simplifies testing - Most SaaS and IT organizations routinely include testing in theirdaily operations. The key is to keep things as basic as you can. Automation has a lotof advantages. The test scripts can be reused when automating test tools.

Quickens the testing procedure - Machines and automated technology operate morequickly than people. This is why we employ them, along with increased accuracy. Your software development cycles are subsequently shortened by this.

Lessens the requirement for human supervision - Tests may be conducted at any timeof day, including overnight. Additionally, when done automatically, this can lessen the possibility of human error.

5.2.6 Selenium Testing

An open-source programme called Selenium automates web browsers. It offers a single interface that enables you to create test scripts in a number of different programming languages, including Ruby, Java, NodeJS, PHP, Perl, Python, and C #. Web application testing for cross-browser compatibility is automated using the Selenium testing tool. Whether they are responsive, progressive, or standard, it is utilized to assure high-quality web apps. Selenium is a free software programme.

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

The project's implementation phase is where the conceptual design is transformed into a functional system. It can be regarded as the most important stage in creating a successful new system since it gives users assurance that the system will operate as intended and be reliable and accurate. User documentation and training are its main concerns. Usually, conversion happens either during or after the user's training. Implementation is the process of turning a newly revised system design into an operational one, and it simply refers to placing a new system design into operation.

The user department now bears the most of the workload, faces the most disruption, and hasthe biggest influence on the current system. If the implementation is not well thought out or managed, confusion and mayhem may result.

Implementation encompasses all of the steps used to switch from the old system to the newone. The new system could be entirely different, take the place of an existing manual or automated system, or it could be modified to work better. A reliable system that satisfies organizational needs must be implemented properly. System implementation refers to the process of actually using the built system. This comprises all the processes involved in switching from the old to the new system. Only after extensive testing and if it is determined that the system is operating in accordance with the standards can it be put into use. The system personnel assess the system's viability. The effort necessary for system analysis anddesign to implement the three key components of education and training, system testing, andchangeover will increase in proportion to how complicated the system being implemented is.

The following tasks are included in the implementation state:
Meticulous planning.

- A system and constraint investigation.
- Development of transitional approaches

6.2 IMPLEMENTATION PROCEDURES

Software implementation refers to the complete installation of the package in its intended environment, as well as to the system's functionality and satisfaction of its intended applications. The software development project is frequently commissioned by someone who will not be using it. People have early reservations about the software, but we must watch out that they do not become more resistant by making sure that:

- The new system's advantages must be known to the active user.
- Their faith in the software is increased.
- The user receives the appropriate instruction so that he feels confident using the application.

Before examining the system, the user must be aware that the server software needs to be running on the server in order to access the results. The actual process won't happen if the server object is not active and functioning on the server.

6.2.1 User Training

The purpose of user training is to get the user ready to test and modify the system. to fulfil the purpose and get the benefits anticipated from a computer-based system, the participantsmust have faith in their ability to fulfil their duties under the new system. Training is more necessary as systems get more complicated. The user learns how to enter data, handle errorwarnings, query the database, call up routines that will generate reports, and execute other important tasks through user training.

6.2.2 Training on the Application Software

The user will need to receive the essential basic training on computer awareness after whichthe new application software will need to be taught to them. This will explain the fundamental principles of how to use the new system, including how the screens work, what kind of help is displayed on them, what kinds of errors are made while entering data, how each entry is validated, and how to change the date that was entered. Then, while imparting the program's training on the application, it should cover the information required by the particular user or group to operate the system or a certain component of the system. Depending on the user group and hierarchical level, this training could be different.

6.2.3 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the softwarecycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In here, I would like to conclude my work. The project entitled '**LinkedNet**' represents a company social networking application. Employers can use the social networking site LinkedNet to find new employees, while job seekers can use it to advance their careers. Users can also share their achievements or memorable moments in this application. It is a great improvement over the manual system. Without looking through newspaper ads, users can quickly find jobs. The system was created and tested with all conceivable examples of data using HTML as the front end and Django as the back end, coupled with SQLite as the database. The system's performance is guaranteed to be effective and able to satisfy all of the user's needs. The performance of the system is provided to be efficient and can meet all the requirements of the user. It provides a fast and efficient web application that can be used by job seekers and employers.

7.1 FUTURE SCOPE

- The web application can be migrated to Android so that the users easily access.
- Implement recommendations by the use of machine learning.
- Search Engine Optimization (SEO) to optimize the search strategy.
- Search users based on location and interest.
- Implement notifications.
- Add comments on post.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

7.1.1 Python: The complete reference python – 20 March 2018 by Martin C. Brown

7.1.2 Let us Python by Kanetkar Yashavant

7.1.3 Reger S Pressman, “Software Engineering”, 2006

7.1.4 Pankaj Jalote “Software Engineering: a precise approach”, 2006

WEBSITES:

7.1.5 https://www.djangoproject.com/

7.1.6 https://www.w3schools.com/Css/css_intro.asp

7.1.7 https://www.geeksforgeeks.org/how-to-move-an-image-with-the-mouse-in-pygame/

7.1.8 https://stackoverflow.com/questions/730207/django-model-set-foreign-key-to-a-fieldofanothermodel#:~:text=You%20need%20to%20use%20%22to_field%22%20in%20%22models.ForeignKey%20%28%29%22,a%20different%20field%2C%20that%20field%20must%20have%20unique%3DTrue.

CHAPTER 9
APPENDIX

9.1 Sample Code

9.1.2Views.py

```
from django.contrib.auth import get_user_model, authenticate
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth
from django.contrib import messages
from django.http import HttpResponse, HttpResponseRedirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth import get_user_model
from .models import Profile, Post, LikePost, FollowersCount, Company
from itertools import chain
import random
from PIL import Image


# Create your views here.

@login_required(login_url='signin')
def index(request):
    user_object = User.objects.get(username=request.user.username)
    user_profile = Profile.objects.get(user=user_object)

    user_following_list = []
    feed = []

    user_following =
FollowersCount.objects.filter(follower=request.user.username)

    for users in user_following:
        user_following_list.append(users.user)

    for usernames in user_following_list:
        feed_lists = Post.objects.filter(user=usernames)
        feed.append(feed_lists)

    feed_list = list(chain(*feed))

    # user suggestion starts
    all_users = User.objects.all()
    user_following_all = []

    for user in user_following:
        user_list = User.objects.get(username=user.user)
        user_following_all.append(user_list)

    new_suggestions_list = [x for x in list(all_users) if (x not in
list(user_following_all))]
    current_user = User.objects.filter(username=request.user.username)
    final_suggestions_list = [x for x in list(new_suggestions_list) if (x not
in list(current_user))]
    random.shuffle(final_suggestions_list)


    username_profile = []
    username_profile_list = []

    for users in final_suggestions_list:
        username_profile.append(users.id)
    for ids in username_profile:
        profile_lists = Profile.objects.filter(id_user=ids)
        username_profile_list.append(profile_lists)

    suggestions_username_profile_list = list(chain(*username_profile_list))

    return render(request, 'index.html', {'user_profile': user_profile,
'posts': feed_list,

'suggestions_username_profile_list': suggestions_username_profile_list[:4]})


@login_required(login_url='signin')
def upload(request):
    if request.method == 'POST':
        user = request.user.username

        if request.method == 'POST':
            # check if attachment file is not empty inside try/except to pass
django error.
            try:
                image = request.FILES["image_upload"]
            except:
                image = None

            # check if uploaded image is valid (for example not video file )
.
            if not image == None:
                try:
                    Image.open(image)
                except:
                    messages.warning(request, 'Sorry, your image is invalid')
                    return redirect('/')

        # image = request.FILES.get('image_upload')
        caption = request.POST['caption']

        new_post = Post.objects.create(user=user, image=image,
caption=caption)
        new_post.save()

        return redirect('/')
    else:
        return redirect('/')


@login_required(login_url='signin')
def search(request):
    user_object = User.objects.get(username=request.user.username)
    user_profile = Profile.objects.get(user=user_object)

    if request.method == 'POST':
        username = request.POST['username']
        username_object = User.objects.filter(username__icontains=username)

        username_profile = []
        username_profile_list = []

        for users in username_object:
            username_profile.append(users.id)

        for ids in username_profile:
            profile_lists = Profile.objects.filter(id_user=ids)
            username_profile_list.append(profile_lists)

        username_profile_list = list(chain(*username_profile_list))
    return render(request, 'search.html', {'user_profile': user_profile,
'username_profile_list': username_profile_list})


@login_required(login_url='signin')
def like_post(request):
    username = request.user.username
    post_id = request.GET.get('post_id')

    post = Post.objects.get(id=post_id)

    like_filter = LikePost.objects.filter(post_id=post_id,
username=username).first()

    if like_filter == None:
        new_like = LikePost.objects.create(post_id=post_id,
username=username)
        new_like.save()
        post.no_of_likes = post.no_of_likes + 1
        post.save()
        return redirect('/')
    else:
        like_filter.delete()
        post.no_of_likes = post.no_of_likes - 1
        post.save()
        return redirect('/')


@login_required(login_url='signin')
def profile(request, pk):
    user_object = User.objects.get(username=pk)
    user_profile = Profile.objects.get(user=user_object)
    user_posts = Post.objects.filter(user=pk)
    user_post_length = len(user_posts)

    follower = request.user.username
    user = pk

    if FollowersCount.objects.filter(follower=follower, user=user).first():
        button_text = 'Unfollow'
    else:
        button_text = 'Follow'

    user_followers = len(FollowersCount.objects.filter(user=pk))
    user_following = len(FollowersCount.objects.filter(follower=pk))

    context = {
        'user_object': user_object,
        'user_profile': user_profile,
        'user_posts': user_posts,
        'user_post_length': user_post_length,
        'button_text': button_text,
        'user_followers': user_followers,
        'user_following': user_following,
    }
    return render(request, 'profile.html', context)
```

```
@login_required(login_url='signin')
def follow(request):
    if request.method == 'POST':
        follower = request.POST['follower']
        user = request.POST['user']

        if FollowersCount.objects.filter(follower=follower,
user=user).first():
            delete_follower = FollowersCount.objects.get(follower=follower,
user=user)
            delete_follower.delete()
            return redirect('/profile/' + user)
        else:
            new_follower = FollowersCount.objects.create(follower=follower,
user=user)
            new_follower.save()
            return redirect('/profile/' + user)
    else:
        return redirect('/')


@login_required(login_url='signin')
def settings(request):
    user_profile = Profile.objects.get(user=request.user)

    if request.method == 'POST':

        if request.FILES.get('image') == None:
            image = user_profile.profileimg
            bio = request.POST['bio']
            location = request.POST['location']
            phone = request.POST['phone']
            tenth = request.POST['tenth']
            twelfth = request.POST['twelfth']
            ug = request.POST['ug']
            pg = request.POST['pg']

            user_profile.puuuurofileimg = image
            user_profile.bio = bio
            user_profile.location = location
            user_profile.phone = phone
            user_profile.tenth = tenth
            user_profile.twelfth = twelfth
            user_profile.ug = ug
            user_profile.pg = pg
            user_profile.save()
        if request.FILES.get('image') != None:

            image = request.FILES.get('image')
            bio = request.POST['bio']
            location = request.POST['location']
            phone = request.POST['phone']

            user_profile.profileimg = image
            user_profile.bio = bio
            user_profile.location = location
            user_profile.phone = phone
            user_profile.tenth = tenth
            user_profile.twelfth = twelfth
            user_profile.ug = ug
            user_profile.pg = pg
            user_profile.save()
        return redirect('/')
    return render(request, 'setting.html', {'user_profile': user_profile})


def signup(request):
    if request.method == 'POST':
        username = request.POST['username']
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        email = request.POST['email']
        password = request.POST['password']
        password2 = request.POST['password2']

        if password == password2:
            if User.objects.filter(email=email).exists():
                messages.info(request, 'Email Taken')
                return redirect('signup')
            elif User.objects.filter(username=username).exists():
                messages.info(request, 'Username Taken')
                return redirect('signup')
            else:
                user = User.objects.create_user(username=username,
first_name=first_name, last_name=last_name, email=email, password=password)
                user.save()

                # log user in and redirect to settings page
                user_login = auth.authenticate(username=username,
password=password)
                auth.login(request, user_login)

                # create a Profile object for the new user
                user_model = User.objects.get(username=username)
                new_profile = Profile.objects.create(user=user_model,
id_user=user_model.id)
                new_profile.save()
                return redirect('settings')
        else:
            messages.info(request, 'Password Not Matching')
            return redirect('signup')

    else:
        return render(request, 'signup.html')


def signin(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = auth.authenticate(username=username, password=password)

        if user is not None:
            auth.login(request, user)
            return redirect('/')
        else:
            messages.info(request, 'Credentials Invalid')
            return redirect('signin')

        company = Company.authenticate(username=username, password=password)
        if company is not None:
            company.login(request, company)
```

```
                return redirect('/')
            else:
                messages.info(request, 'Credentials Invalid')
                return redirect('signin')

    else:
        return render(request, 'signin.html')


@login_required(login_url='signin')
def logout(request):
    auth.logout(request)
    return redirect('signin')


def csignup(request):
    if request.method == 'POST':
        username = request.POST['username']
        cname = request.POST['cname']
        location = request.POST['location']
        website = request.POST['website']
        email = request.POST['email']
        password = request.POST['password']
        password2 = request.POST['password2']

        if password == password2:
            if Company.objects.filter(email=email).exists():
                messages.info(request, 'Email Taken')
                return redirect('csignup')
            elif Company.objects.filter(username=username).exists():
                messages.info(request, 'Username Taken')
                return redirect('csignup')
            else:
                company = Company(username=username, cname=cname,
location=location, website=website, email=email, password=password)
                company.save()
                return redirect('signin')
        else:
            messages.info(request, 'Password Not Matching')
            return redirect('csignup')

    else:
        return render(request, 'csignup.html')
```

9.2 Screen Shots

9.2.1 Login.html

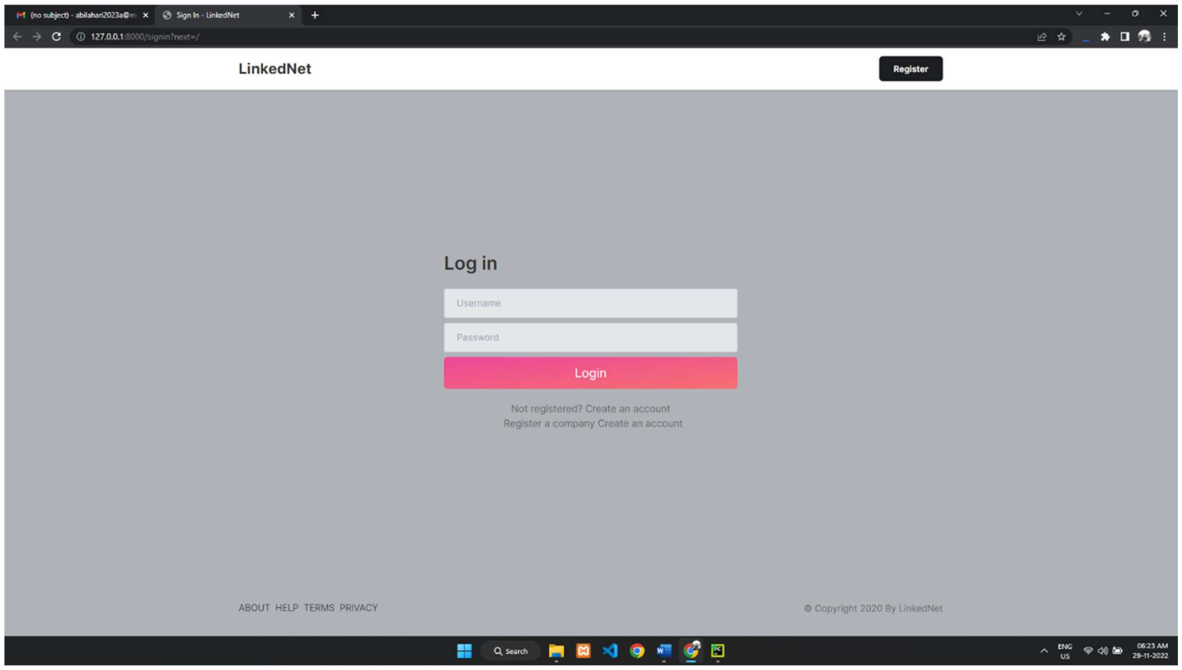

LinkedNet
Register
Log in
Username
Password
Login
Not registered? Create an account
Register a company Create an account
ABOUT HELP TERMS PRIVACY
© Copyright 2020 By LinkedNet

Figure 9.2.1

9.2.2 Registration.html

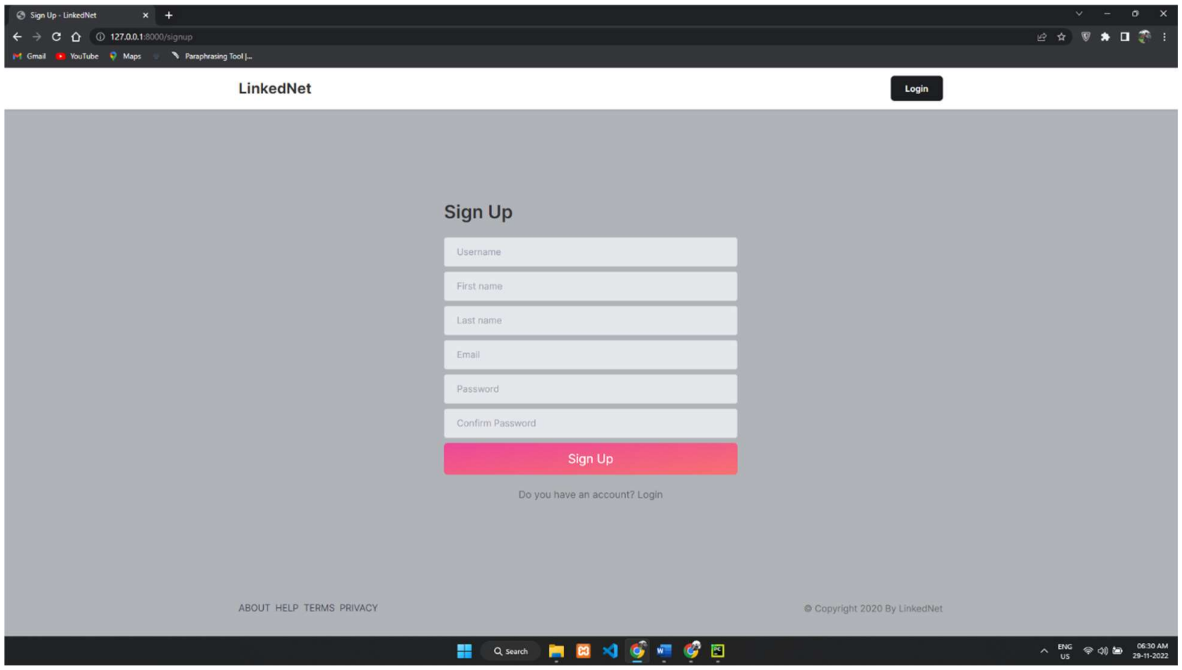


Figure 9.**2**.2

9.2.3 Home.html

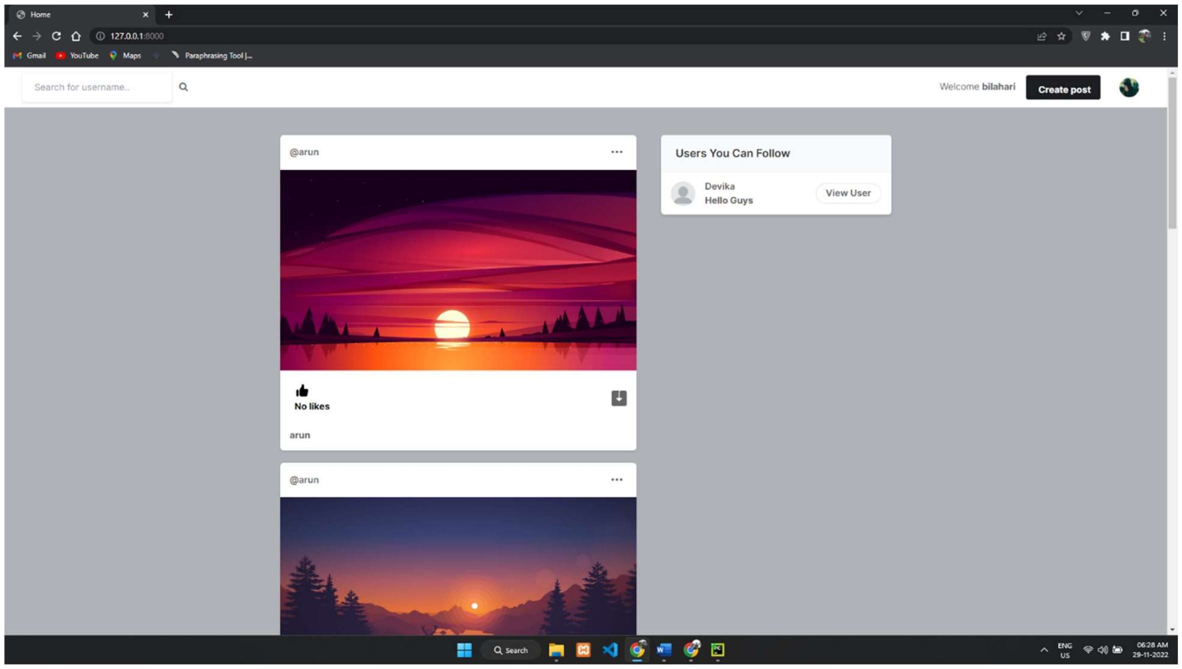


Figure 9.2.3

9.2.4 Profile.html

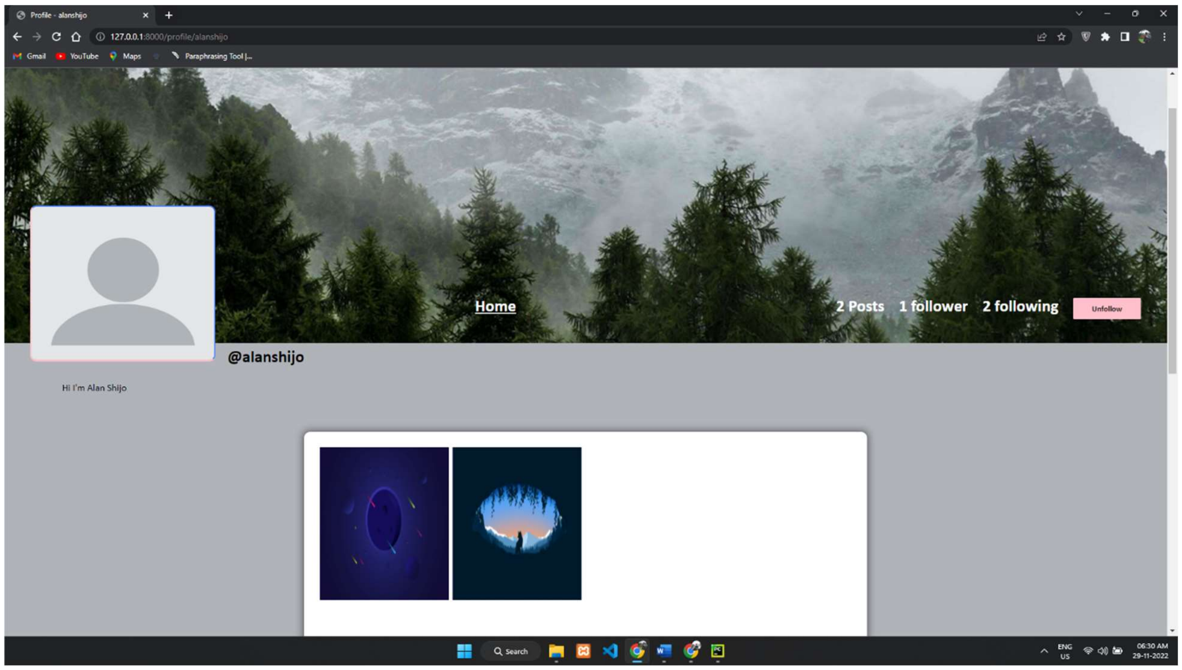
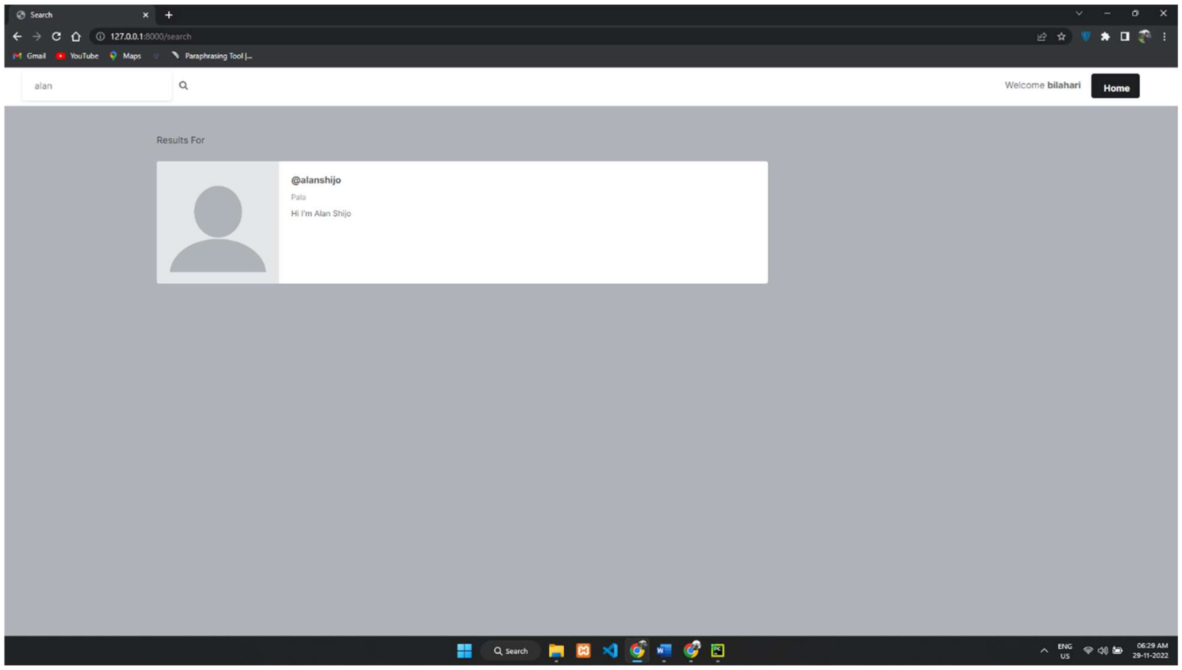


Figure 9.2.4

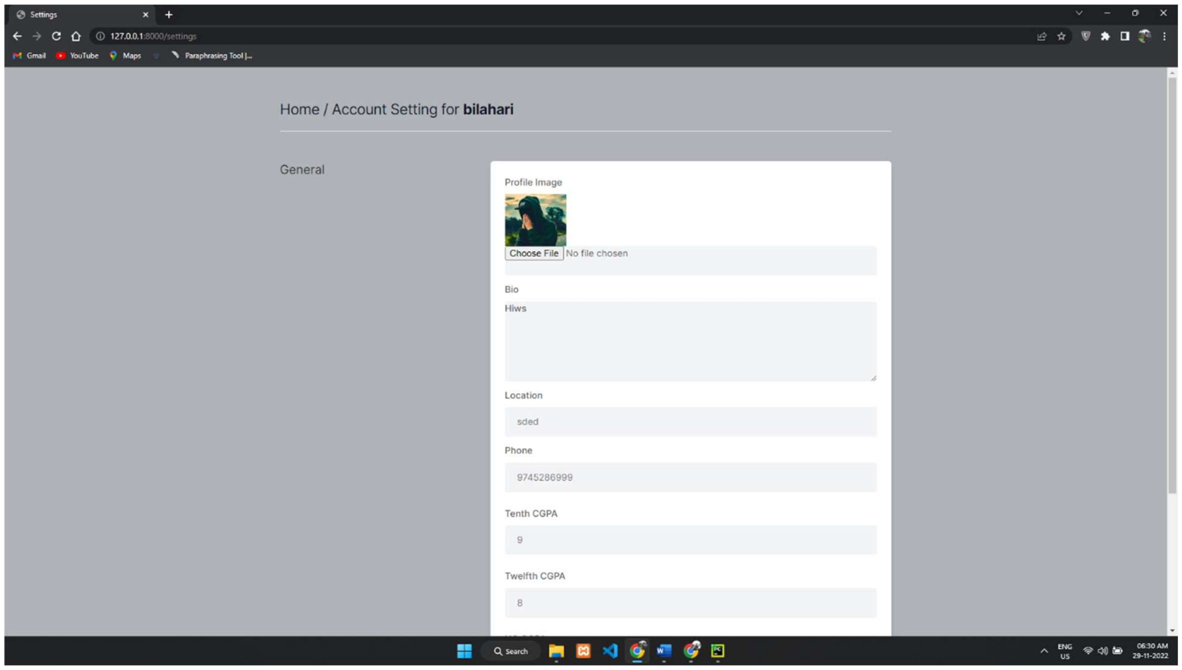
9.2.5 Search.html



@alanshijo
Pala
Hi I'm Alan Shijo

Figure 9.2.5

9.2.6Settings.html



Home / Account Setting for **bilahari**

Profile Image

Bio
Hiws

Location
sded

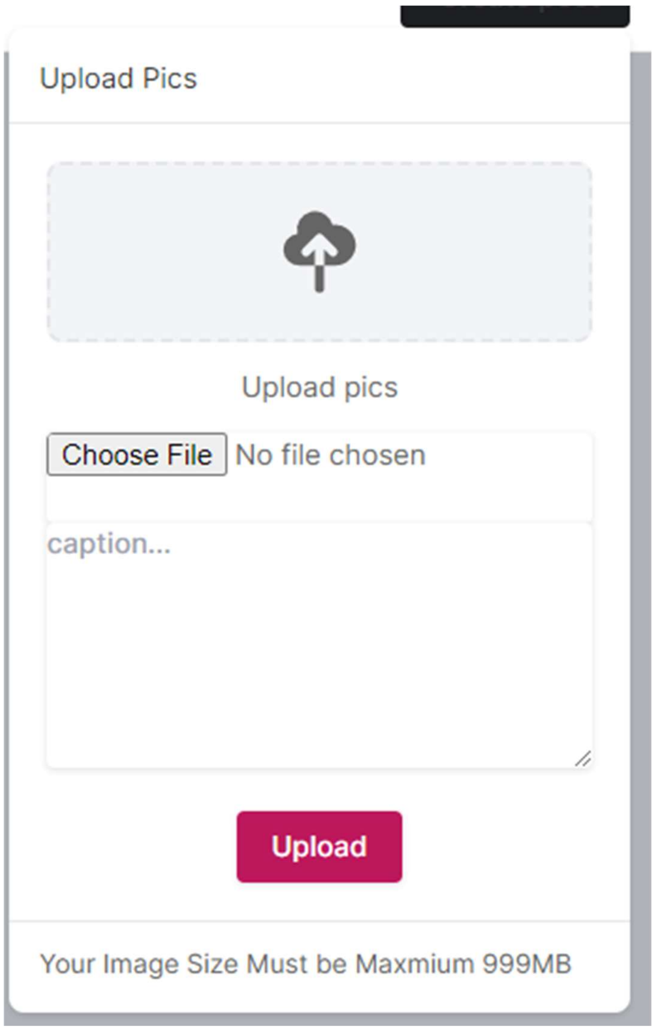
Phone
9745286999

Tenth CGPA
9

Twelfth CGPA
8

Figure 9.2.6

9.2.7 upload_post.html


Upload Pics

Upload pics

Choose File No file chosen

caption...

Upload

Your Image Size Must be Maxmium 999MB

Figure 9.2.7