RAMANUJAN COLLEGE

(UNIVERSITY OF DELHI)

BSc.(H) COMPUTER SCIENCE
6th SEMESTER

TEACHER
CONNECT

"TEACHER CONNECT"

Submitted by :-

1. Prerna(20020570025)
2. Sumeg Yogram Sharnagat(20020570034)
3. Mohd Shohel(20020570022)
4. M D Kaif(20020570019)

SOFTWARE REQUIREMENTS AND SPECIFICATIONS

Introduction:

Purpose

This software will provide teachers of various colleges from different departments to connect on a single platform. Teachers of Delhi University using it can lookout for teachers of similar interest and background for various collaborations for research and development, to connect and share their ideas with each other and facility for discussion . To circulate various information and notifications and various other messages.

Existing system

In existing system, if a teachers want to connect to a teacher of a certain background, first of all he has to signup up for linkedin and then he needs to search with the filter of that specific background and then look for a teacher form delhi university he may or may not find even if there may such a teacher exist.
There may exist various difficulties using such a way if all the teachers are not on the same platform. So by bringing them together using software, they solve these kinds of hectic problems of searching teachers.

Problem statement

The objective is to provide a platform where teachers of similar universities can find and interact with each other. To collaborate and publish joint venture research papers to able contact for discussion and forward information and viewpoints and opinions.

Document Conventions

This document features some terminology which readers may be unfamiliar with

SRS: Software Requirement Specification

End: This stands for the JSP pages that a user will see when (s) he will access the application through the web.

Admin:A user which is hired to add college,department and teachers details.

Entry: A data unit stored to the database

Sign up: Creating New User

Log in: Logging in Existing User

Intended Audience and Reading Suggestions

The intended readers of this document are current and future developers working on and the sponsors of the project.

Scope

The scope of the software is to provide a platform for teachers of different colleges and departments under Delhi University to interact with each other, collaborate on research projects, and share their ideas and views. The software will have the following features:

User Registration and Login: Teachers will be able to register themselves on the platform by providing their basic details such as name, email, and contact number. Once registered, they can log in using their credentials.

Teacher Profile: Each teacher will have a profile page that will include their personal details, educational qualifications, research interests, and publications.

Search and Filter: The software will allow users to search and filter other teachers based on their department, research interests, and location.

Collaboration and Discussion: The platform will provide a facility for users to collaborate and discuss their research projects with other teachers.

<u>Notifications</u>: Users will receive notifications about new research projects, events, and discussions on the platform.

<u>SuperAdmin/Admin Panel</u>: An admin panel will be provided to manage the details of colleges, departments, and teachers

<u>Security</u>: The software will be secure and ensure the confidentiality of user data.

Product Perspective

The software will be a web-based application that will provide a platform for teachers of different colleges and departments under Delhi University to interact with each other, collaborate on research projects, and share their ideas and views. The software will be designed to meet the needs of teachers who want to connect with their colleagues and collaborate on research projects.

The software will be developed using Python, Django, and Sqlite technologies. It will be compatible with different browsers and devices, and it will be scalable to accommodate future growth and new features. The software will have a user-friendly interface and easy navigation for a better user experience.

The software will have two main components: the **Front end** and the **Back end**. The front end will be the Django webpage that the users will see when they access the application through the web. The back end will be the database and server-side scripting that will handle the data and application logic.

The software will be developed with security in mind to ensure the confidentiality of user data and prevent unauthorized access. The users will be required to register themselves on the platform by providing their basic details such as name, email, and contact number. Once registered, they can log in using their credentials.

The software will provide a facility for users to collaborate and discuss their research projects with other teachers on the platform. The users will be able to search and filter other teachers based on their department, research interests, and colleges. The software will send notifications to users about new research projects, events, and discussions on the platform.

An admin panel will be provided to manage the details of colleges, departments, and teachers. The admin panel will be used by an admin user who will be responsible for adding and managing the details of colleges, departments, and teachers.

Overall, the software will be a comprehensive platform for teachers to connect, collaborate, and share their ideas and views. It will provide a valuable resource for teachers who want to enhance their research and development activities and contribute to the academic community.

Functions of the Product

- User registration and login: The software will allow users to create a new account by registering themselves on the platform using their name, email, and contact number. Users will be able to log in to the platform using their credentials.
- User profile management: Users will be able to manage their profile by updating their personal details, academic qualifications, research interests, and other relevant information.
- Search and filter: Users will be able to search and filter other teachers based on their department, research interests, and location.
- Collaboration: Users will be able to collaborate with other teachers on the platform for research projects and development activities.
- Discussion forum: The software will provide a discussion forum for users to discuss various academic topics, share their opinions and views, and get feedback from their colleagues.
- Notification and messaging: The software will send notifications and messages to users about new research projects, events, and discussions on the platform.
- Admin panel: An admin panel will be provided to manage the details of colleges, departments, and teachers. The admin user will be able to add and manage the details of colleges, departments, and teachers.
- Information and notification dissemination: The software will allow the admin user to circulate various information and notifications to the users.
- Research paper publishing: The software will provide a facility for users to publish joint venture research papers with other teachers on the platform.
- User feedback: The software will allow users to provide feedback on the platform's features and functionalities to improve the user experience.

PROCESS MODEL USED

The process model that will be used for the development of the software will be the Agile Scrum model.

Agile Scrum is a popular iterative and incremental process model used for developing software. It is a flexible and adaptive approach that allows for continuous delivery of functional software throughout the development process. The Agile Scrum model is based on the principles of collaboration, adaptability, and responsiveness.
In Agile Scrum, the development team works in short cycles called sprints. Each sprint typically lasts for two to four weeks, during which the team works on a set of prioritized features or requirements. The development team works closely with the product owner and other stakeholders to define the requirements and prioritize the work.

The Agile Scrum process model involves the following key roles:

1. Product Owner: The product owner is responsible for defining the product vision, prioritizing the product backlog, and communicating the product requirements to the development team.
2. Scrum Master: The scrum master is responsible for facilitating the scrum process, removing any impediments that hinder the team's progress, and ensuring that the team adheres to the Scrum principles.
3. Development Team: The development team is responsible for designing, coding, testing, and delivering the software product incrementally and iteratively.
   In the Agile Scrum model, the development team works on a set of prioritized features or requirements, which are called the product backlog. The product backlog is refined and prioritized before the start of each sprint. During the sprint, the development team works on the top-priority items from the product backlog and delivers a potentially shippable product increment at the end of each sprint.

FEASIBILITY OF THE PRODUCT

To assess the feasibility of the product, we need to consider the following factors:

1. Technical feasibility: Is the technology available and suitable to develop the software? Does the development team have the required skills and

resources to develop and maintain the software? The technical feasibility of the product depends on the availability of the required hardware, software, and technical expertise.

2. Economic feasibility: Is the product economically viable? Will the cost of development, maintenance, and support be lower than the potential benefits and revenue generated by the product? The economic feasibility of the product depends on the market demand, competition, pricing strategy, and revenue model.
3. Operational feasibility: Will the software be easy to use, maintain, and support? Will the users be able to adapt to the new system quickly? The operational feasibility of the product depends on the usability, reliability, and scalability of the software.
4. Legal and regulatory feasibility: Does the product comply with the legal and regulatory requirements? Are there any copyright, patent, or trademark issues that need to be addressed? The legal and regulatory feasibility of the product depends on the local laws and regulations governing the use and distribution of software.

Based on the above factors, we can conclude that the proposed software product is feasible. The technical feasibility of the product is high as the required technology and expertise are readily available. The economic feasibility of the product is also high as there is a market demand for such a platform, and the revenue can be generated through subscriptions or other means. The operational feasibility of the product is good as the software will be designed with a user-friendly interface and robust features. The legal and regulatory feasibility will be ensured by complying with the relevant laws and regulations.
Therefore, we can conclude that the proposed software product is feasible and can be developed with a reasonable degree of certainty.

USER CLASSES AND CHARACTERISTICS

The proposed software product will have the following user classes:

1. Teachers: This user class will include teachers from various colleges and departments of Delhi University who will use the software to connect with each other, collaborate on research projects, and share their ideas and views.
2. Admin: This user class will include the administrator who will be responsible for managing the software, adding new college and department details, and ensuring the smooth functioning of the platform.
3. Sub-admins - This user class has the responsibility of managing the teacher list for a specific college under the University of Delhi. They can add teachers from different departments and modify their profiles.

4. Super-admin - This user class has complete access to the software and all of its components. They have the authority to oversee the entire project and make any necessary changes.

The following are the characteristics of each user class:

1. Teachers: Teachers are the primary users of the software product. They will have different levels of technical expertise, and hence the software should be designed with a user-friendly interface that is easy to use and understand. Teachers will be looking for specific features such as the ability to search for teachers with similar interests and backgrounds, connect with other teachers, share their research papers, and participate in discussions.
2. Admin: The administrator is responsible for managing the software, adding new college and department details, and ensuring the smooth functioning of the platform. The administrator should have strong technical skills and knowledge of software development and management. They will be looking for features that allow them to manage the software effectively and efficiently.
3. Sub-admin:Basic technical skills to navigate the software interface and manage the teacher list. Access to a computer or mobile device with an internet connection. Familiarity with the policies and procedures of their respective college
4. Super-admin:Advanced technical skills and knowledge of web development and software management. Access to a computer or mobile device with an internet connection. In-depth knowledge of the policies and procedures of the University of Delhi. Ability to oversee the entire project, manage the database, and troubleshoot technical issues as needed

All user classes will require secure access to the software platform and its features. The software should be designed to ensure data security, privacy, and confidentiality. The software should also be scalable to accommodate a growing user base and adaptable to meet the changing needs and requirements of the users.

Operating Environment

The proposed software product will operate in an online environment, where it will be accessible through a web browser. The software will be hosted on a server and accessed by users via the internet. The operational environment of the software will require the following components:

1. Server: The software will require a server to host the application and store the database. The server should be reliable, scalable, and secure.

2. Internet Connection: The users will access the software via the internet, and hence a reliable and stable internet connection is essential.

3. Web Browser: The software will be accessed through a web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge. The web browser should be updated and compatible with the latest web standards.

4. Operating System: The software will be compatible with various operating systems such as Windows, macOS, and Linux. The software should be tested and optimized for each operating system.

5. Database: The software will store data in a database, which should be scalable and secure. The database should be optimized for efficient data retrieval and storage.

<u>Design and Implementation Constraints</u>

Design and implementation constraints refer to the limitations that can affect the design and development of the software product. Some of the design and implementation constraints that need to be considered for this project include:

1. <u>Technology constraints</u>: The software product must be developed using specific technologies such as Python, Django and SQLite. The development team should have sufficient knowledge and experience in these technologies to ensure that the software is developed efficiently.
2. <u>Time constraints</u>: The software product must be developed within a specific timeframe. The development team must ensure that the project is completed within the given timeline while meeting all the requirements and specifications.
3. <u>Resource constraints</u>: The development team may have limited resources, such as hardware, software, and human resources. They should ensure that the project is developed within the available resources.
4. <u>Security constraints</u>: The software product must be designed to ensure the security of user data and information. It should be protected against unauthorized access, hacking, and other security threats.

5. Compatibility constraints: The software product must be compatible with various operating systems, web browsers, and devices. It should be tested and optimized for each platform to ensure that it works correctly on all devices.
6. Usability constraints: The software product must be user-friendly, easy to use, and navigate. The development team should ensure that the software interface is intuitive and easy to understand, even for users with limited technical knowledge.
7. Performance constraints: The software product must be designed to perform efficiently, even when there is a large number of users accessing the platform simultaneously. The software should be optimized for speed, efficiency, and reliability.

To overcome these constraints, the development team should carefully plan and execute the project, utilize the available resources efficiently, and ensure that the software product meets all the requirements and specifications. They should also regularly test and maintain the software to ensure that it performs optimally and is compatible with the latest technologies and operating systems.

User Documentation

User documentation refers to the instructional material provided to the users of the software product. It should be designed to help users understand the software, its features, and how to use it effectively. The user documentation for this software product should include the following:

1. User manual: The user manual should provide a detailed description of the software product, its features, and how to use it. It should be written in a clear and concise language that is easy to understand, even for users with limited technical knowledge. The user manual should include step-by-step instructions for common tasks, such as creating an account, searching for other users, and sending messages.

2. FAQs: The FAQs section should provide answers to common questions that users may have about the software product. It should be organized into categories, such as account creation, messaging, and collaboration, to make it easy for users to find the information they need.

3. Video tutorials: Video tutorials can be provided to help users understand the software product better. They can provide a visual demonstration of how to use the software, making it easier for users to follow along and learn.

4. Helpdesk support: A helpdesk support system can be set up to provide users with assistance and support. Users can submit support tickets, which will be handled by a support team that can provide prompt and efficient assistance.

The user documentation should be provided in a variety of formats, such as PDF, online help, and video tutorials, to accommodate different learning styles and preferences. It should be regularly updated and maintained to ensure that it remains relevant and accurate. By providing comprehensive user documentation, users can learn how to use the software effectively and make the most out of its features.

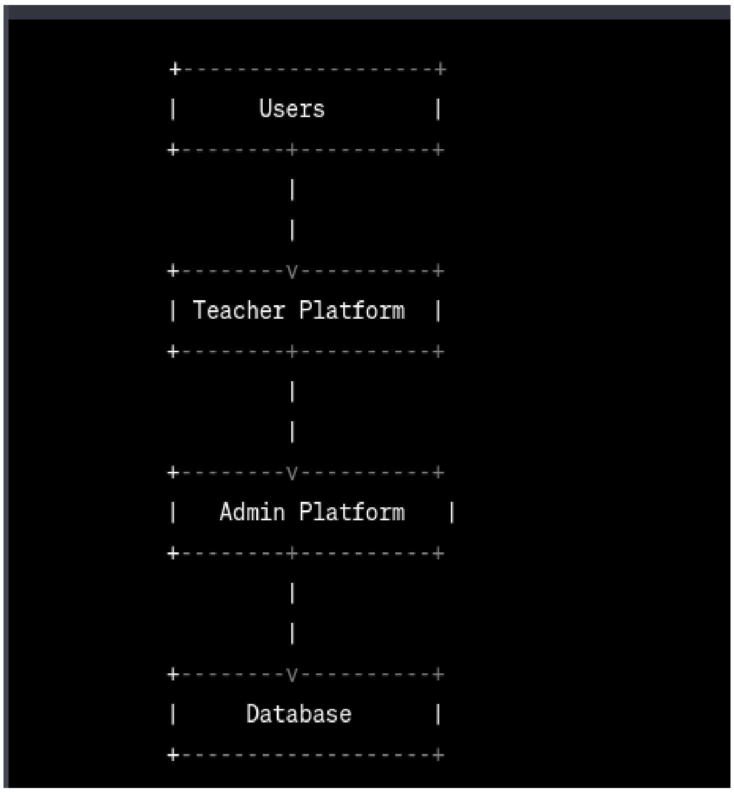
Assumptions

1. The users have basic computer knowledge and internet access.
2. The users will provide accurate and valid information during registration.
3. The users will follow the terms and conditions of the platform.
4. The users will not use the platform for any illegal or unethical activities.

DATA FLOW DIAGRAM

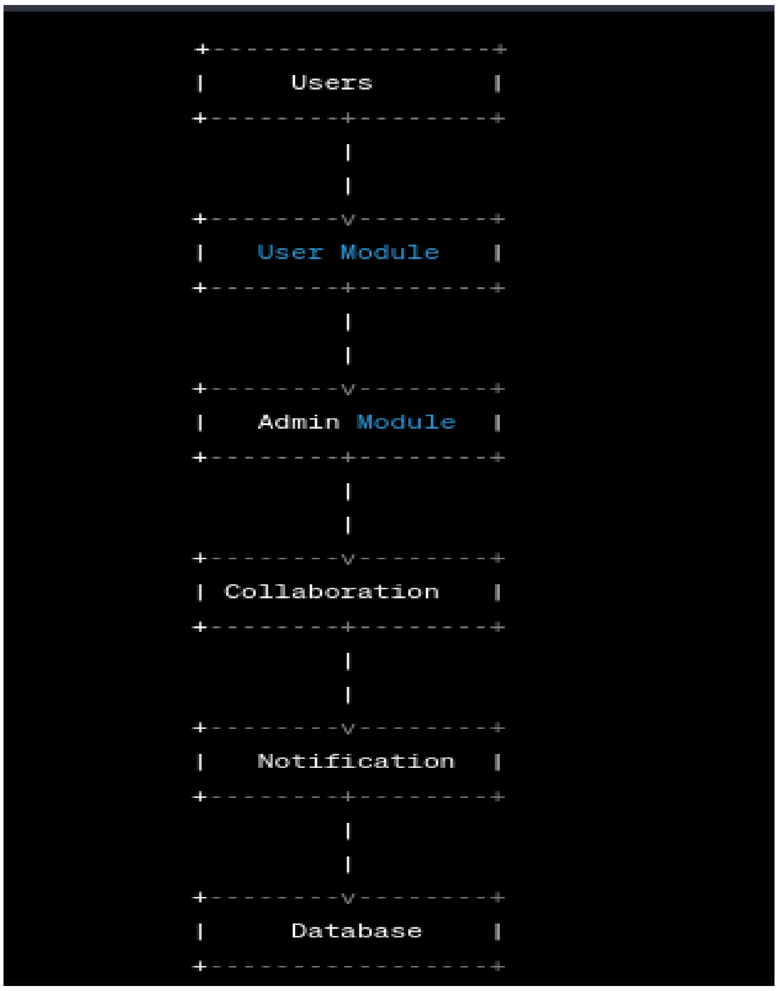
A Data Flow Diagram (DFD) is a visual representation of the flow of data through a system. It shows the inputs, outputs, and processes involved in the system. Here's an example of a DFD for the teacher collaboration software:

Level 0 DFD:



```
+-------------------+
|      Users        |
+--------+----------+
         |
         |
+--------v----------+
| Teacher Platform  |
+--------+----------+
         |
         |
+--------v----------+
|   Admin Platform   |
+--------+----------+
         |
         |
+--------v----------+
|     Database      |
+-------------------+
```

Level 1 DFD:



```
+-----------------+
|      Users      |
+--------+--------+
         |
         |
+--------v--------+
|   User Module   |
+--------+--------+
         |
         |
+--------v--------+
|   Admin Module  |
+--------+--------+
         |
         |
+--------v--------+
| Collaboration   |
+--------+--------+
         |
         |
+--------v--------+
|   Notification  |
+--------+--------+
         |
         |
+--------v--------+
|     Database    |
+-----------------+
```

The Level 1 DFD shows the main modules of the software product:

1. User Module: This module allows users to sign up, log in, search for other users, and collaborate on research projects. It includes features such as messaging, discussion forums, and document sharing.
2. Admin Module: This module allows the administrator to manage user accounts, add or remove users, and monitor the usage of the software.
3. Collaboration: This module allows users to collaborate on research projects, share documents, and exchange ideas.
4. Notification: This module sends notifications to users about new messages, updates, and changes in the system.

The database stores all the data related to user accounts, collaboration projects, and notifications. The DFD shows that data flows from the users to the User Module and Admin Module, and then to the Collaboration and Notification modules. The Collaboration module sends data to the database, which is then used to update the system and send notifications to users. Overall, the DFD provides a clear and concise representation of the data flow and processes involved in the software product.

Data Dictionary

A data dictionary is a reference document that describes the data elements used in a software system. Here's an example data dictionary for the teacher collaboration software:

This data dictionary provides a comprehensive list of the data elements used in the teacher collaboration software. It includes the names of the tables, the names of the fields, the data types and lengths of the fields, and a brief description of each field. The data dictionary serves as a reference for developers and users of the software to understand the structure of the database and the data elements used in the system

Users table

Field Name	Data Type	Length	Description
User ID	Integer	10	Unique identifier for each user
First Name	Varchar	50	First name of the user
Last Name	Varchar	50	Last name of the user
Email	Varchar	100	Email address of the user
Password	Varchar	20	Encrypted password of the user
Department	Varchar	50	Department to which the user belongs
University	Varchar	50	University to which the user belongs
Interests	Varchar	255	Areas of interest for the user
Description	Text		Short description of the user
Image	Blob		Profile image of the user

Collaboration table

Field Name	Data Type	Length	Description
Project ID	Integer	10	Unique identifier for each collaboration project
Title	Varchar	100	Title of the collaboration project

Field Name	Data Type	Length	Description
Description	Text		Description of the collaboration project
Participants	Varchar	255	List of users participating in the project
Start Date	Date		Start date of the project
End Date	Date		End date of the project
Status	Varchar	20	Status of the project (in progress, completed, etc.)
Files	Blob		Files associated with the project

Notification table

Field Name	Data Type	Length	Description
Notification ID	Integer	10	Unique identifier for each notification
User ID	Integer	10	User ID of the recipient of the notification
Message	Text		Message contained in the notification
Timestamp	Datetime		Date and time when the notification was sent
Status	Varchar	20	Status of the notification (read, unread, deleted, etc.)

System Features

1. User Registration: The software allows new users to register by providing basic personal and professional information, such as their name, email address, university, department, and areas of interest.
2. User Login: Registered users can log in to the software by providing their email address and password.
3. User Profile: Each user has a profile page that displays their personal and professional information, including their name, photo, university, department, and areas of interest. Users can also update their profile information as needed.
4. Collaboration Projects: Users can create new collaboration projects and invite other users to participate. Each project has a title, description, start and end dates, and a list of participants. Users can upload and share files associated with the project.

5. Discussion Forum: The software provides a discussion forum where users can start new discussion threads, post messages, and respond to messages from other users.
6. Notifications: Users receive notifications when they are invited to participate in a new project, when a project they are participating in is updated, and when they receive new messages in the discussion forum.
7. Search: The software includes a search feature that allows users to search for other users based on their university, department, and areas of interest.
8. Admin Panel: The admin panel allows a designated administrator to manage user accounts, create new departments and universities, and monitor user activity.
9. Privacy and Security: The software includes measures to protect user data and ensure the privacy of user communications.

These features are just a starting point and may be expanded or customized based on the specific needs of the user community.

External Interface Requirements

User Interfaces

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

The protocol used shall be HTTP. The Port number

used will be 80. There shall be a logical address of the

system in IPv4 format.

Hardware Interfaces

Laptop/Desktop PC

Purpose of this is to give information when Customers ask information about the operator, medicine available lab tests etc. To perform such Action it needs a very efficient computer otherwise due to thatreason customers have to wait for a long time to get what they ask for.

Laser Printer (B/W)

Simply this device is for printing customer's info etc.

Wi-Fi router

Wi-Fi router is used for internetwork operations simply data transmission from pc to server.

Software Interfaces

- Python: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application

Development, as well as for use as a scripting or glue language to connect existing components together.

- Django: Django is a Python framework that makes it easier to create web sites using Python.Django takes care of the difficult stuff so that you can concentrate on building your web applications.
- Visual Studio Code: IDE for development.
- SQLite server : Database connectivity and management
- OS Windows or Ubuntu: Very user friendly and common OS

Communications Interfaces

- NIC (Network Interface Card) – It is a computer hardware component that allows computers to connect to a network. NICs may be used for both wired and wireless connections.
- CAT 5 network cable- for high signal integrity
- TCP/IP protocol- Internet service provider to access and share information over the Internet
- Ethernet Communications Interface- Ethernet is a frame-based computer network technology for local area networks (LANs)
- Ubiquitous, easy to set up and easy to use. Low cost and high data transmission rates

Non-Functional Requirements

1. Performance: The software shall be able to handle a large number of users and data without any performance issues.
2. Security: The software shall ensure the confidentiality of user data and prevent unauthorized access.
3. User Experience: The software shall have a user-friendly interface and easy navigation for a better user experience.
4. Compatibility: The software shall be compatible with different browsers and devices.
5. Availability: The software shall be available 24/7 without any downtime for maintenance.
6. Scalability: The software shall be scalable to accommodate future growth and new features.

Safety Requirements

if there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed-up log, up to the time of failure. All the administrative and data entry operators have unique logins so the system can understand who is login in to the system right now no intruders allowed except system administrators. Nobody cannot change records and valuable data.

Software Quality Attributes

AVAILABILITY: The system shall be available all the time.

CORRECTNESS: bug free software which fulfills the correct need/requirements of the client.

MAINTAINABILITY: The ability to maintain, modify information and update fix problems of the system.

USABILITY: software can be used again and again without distortion.

ACCESSIBILITY: Administrator and many other users can access the system but the access level is controlled for each user according to their work scope.

ACCURACY: The reliability of the information/output. Can depend/be sure of the outcome.

STABILITY: The system outcome/output won’t change time to time. Same output Will always be given for a given input.

Security Requirements

Want take the responsibility of failures due to hardware malfunctioning.

1. Warranty period of maintaining the software would be one year.
2. Additional payments will be analyzed and charged for further maintenance
3. If any error occurs due to a user’s improper use. Warranty will not be allocated to it.

No money back returns for the software.

ESTIMATION AND SCHEDULING

PROJECT METRICS

Project metrics are used to control and coordinate software engineering processes to improve the quality of the software to be produced. Project specific metrics provide indication of productivity and insight into the technical activities. Adapt project workflow and technical activities and code.

SIZE ESTIMATION

Function-Based Metrics

The function point (FP) metric can be used effectively as a means for measuring the functionality delivered by a system.4 Using historical data, the FP metric can then be used to (1) estimate the cost or effort required to design, code, and test the software; (2) predict the number of errors that will be encountered during testing; and (3) forecast the number of components and/or the number of projected source lines in the implemented system. Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and qualitative assessments of software
complexity. Information domain values are defined in the following manner:

Number of external inputs (EIs). Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately.

Number of external outputs (EOs). Each external output is derived data within the application that provides information to the user. In this context external output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.

Number of external inquiries (EQs). An external inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an ILF).

Number of internal logical files (ILFs). Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs.

Number of external interface files (EIFs). Each external interface file is a logical grouping of data that resides external to the application but provides information that may be of use to the application.
Once these data have been collected, the table in Figure 23.1 is completed and a complexity value is associated with each count. Organizations that use function point

Methods develop criteria for determining whether a particular entry is simple, average,or complex. Nonetheless, the determination of complexity is somewhat subjective.

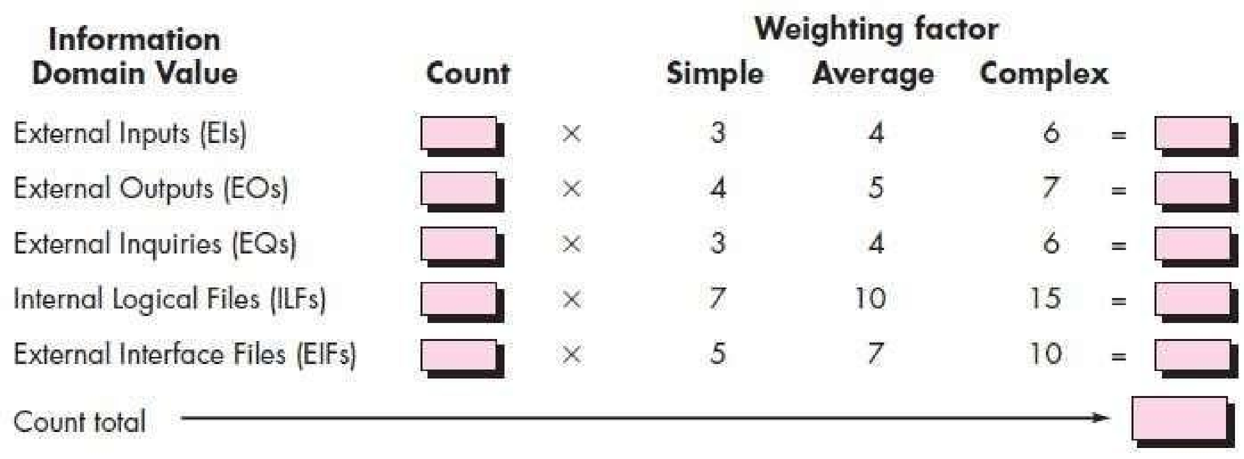
To compute function points (FP), the following relationship is used:

Calculation of **complexity adjustment values**:

	Grade value
Does the system require reliable backup and recovery?	5
Are data communications required?	4
Are there distributed processing functions?	3
Is performance critical?	5
Will the system run in an existing, heavily utilized operational environment?	4
Does the system require online data entry?	3
Does the on-line data entry require the input transaction to be built over multiple screens or operations?	3
Are the master files updated online?	3
Are the inputs, outputs, inquiries complex?	2
Is the internal processing complex?	3
Is the code designed to be reusable?	4
Are conversion and installation included in the design?	3

Is the system designed for multiple installations in different organizations?	3
Is the application designed to facilitate change and ease of use by the user?	5
∑F =	50

Calculation of Function point for the current software:



Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)		×	3	4	6	=	
External Outputs (EOs)		×	4	5	7	=	
External Inquiries (EQs)		×	3	4	6	=	
Internal Logical Files (ILFs)		×	7	10	15	=	
External Interface Files (EIFs)		×	5	7	10	=	
Count total							

Measurement parameter	**Count**	**Weighting Factor**	**Weighting count**
Number of user inputs	6	4	24
Number of user outputs	4	5	20

Number of user inquiries	1	4	4
Number of files	3	10	30
Number of external interfaces	0	7	0
Count total=			78

$$\text{Function point} = \text{Total count} * (0.65 + 0.01 * (\textstyle\sum F))$$

$$= 78 * (0.65 + 0.01 * 50)$$

$$= 89.7$$

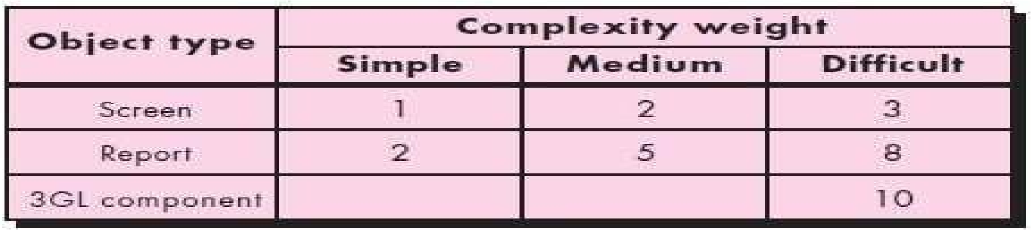
Once function point has been calculated it can be used to normalize measures for software quality, productivity and otherattributes such as:

- Errors per FP
- Defects per FP
- $ per FP

<u>Cost Estimation</u>

The COCOMO II Model

In his classic book on “software engineering economics,” Barry Boehm [Boe81] introduced a hierarchy of software estimation models bearing the name COCOMO, for Constructive Cost Model.The original COCOMO model became one of the most widely used and discussed software cost estimation models in



Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

the industry. It has evolved into a more comprehensive estimation model, called COCOMO II [Boe00]. Like its predecessor, COCOMO II is actually a hierarchy of estimation models that address the following areas:

- Application composition model. Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.

- Early design stage model. Used once requirements have been stabilized and basic software architecture has been established.
- Post-architecture-stage model. Used during the construction of the software.

Like all estimation models for software, the COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy: object points, function points, and lines of source code.

The COCOMO II application composition model uses object points and is illustrated in the following paragraphs. It should be noted that other, more sophisticated estimation models (using FPand KLOC) are also available as part of COCOMO II.Like function points, the object point is an indirect software measure that is computed using counts of the number of (1) screens (at the user interface), (2) reports, and (3) components likely to be required to build the application. Each objectinstance (e.g., a screen or report) is classified into one of three complexity levels (i.e.,simple,medium, or difficult) using criteria suggested by Boehm [Boe96]. In essence,complexity is a function of the number and source of the client and server data tables that are required to generate the screen or report and the number of views or sections presented as part of the screen or report.

$$\text{NOP} = (\text{object points}) \times [(100 - \%\text{reuse})/100]$$

Once complexity is determined, the number of screens, reports, and components are weighted according to the table illustrated in Figure . The object point count is then determined by multiplying the original number of object instances by the weighting factor in the figure and summing to obtain a total object point count. When component-based development or general software reuse is to be applied, the percent of reuse (%reuse) is estimated and the object point countis adjusted:

where NOP is defined as new object points.

To derive an estimate of effort based on the computed NOP value, a “productivity rate” must be derived. Figure B presents the productivity rate

$$\text{PROD} = \frac{\text{NOP}}{\text{person-month}}$$

for different levels of developer experience and development environment maturity.Once the productivity rate has been determined, an estimate of project effort is computed using

$$\text{Estimated effort} = \frac{\text{NOP}}{\text{PROD}}$$

In more advanced COCOMO II models,12 a variety of scale factors, cost drivers, and adjustment procedures are required.

PRODUCTIVITY RATE FOR OBJECT POINT COUNTS

Developer's experience/capability	Very low	Low	Nominal	High	Very high
Environment maturity/capability	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

COST ESTIMATION FOR THIS PROJECT

SCREENS

1 LOGIN TYPE
2. LOGIN
3.REGISTRATION
4.ABOUT US
5.UPDATE COLLEGE DETAILS
6.DELETE COLLEGE DETAILS
7.TEACHER MAIL OPTION
8.VIEW TEACHERS
9. VIEW COLLEGES
10.UPDATE TEACHER
11.VIEW DEPARTMENT BY ADMIN

TOTAL SCREENS= 11

REPORTS

1. REGISTERED SUCCESSFULLY
2. DETAILS SUCCESSFULLY UPDATED
3. COLLEGE SUCCESSFULLY MADE
4. USER CAN MAIL

TOTAL REPORTS =4

3 GL MODULES

1.PYTHON
TOTAL 3GL MODULES =1 CONSIDERING ALL OF ABOVE HAVE AVERAGE COMPLEXITY WE GET OBJECT POINT= 11*2+4*5+1*10
OP=52

NOP=OBJECT POINT COUNT–(1-%REUSE/100) NOP=52*(1-0.50)
NOP=26

NOW TAKING THE DEVELOPER EXPERIENCE AND SKILLS AS NOMINAL WE HAVE PRODUCTIVITY=13
EFFORT= NOP/PRODUCTIVITY

EFFORT= 26/13

EFFORT=2
EFFORT= 2 PERSON/MONTH

SCHEDULING

Scheduling of a software project does not differ greatly from scheduling of any multi task engineering effort. Therefore, generalized project scheduling tools and techniques
Can be applied with little modification for software projects.

Program evaluation and review technique (PERT) and the critical path method (CPM)are two project scheduling methods that can be applied to software development. Both techniques are driven by information already developed in earlier project planning activities: estimates of effort, a decomposition of the
product function, the selection of the appropriate process model and task set, and decomposition of the tasks that are selected.
Interdependencies among tasks may be defined using a task network. Tasks, sometimes called the project work breakdown structure (WBS), are defined for the product as a whole or for individual functions.
Both PERT and CPM provide quantitative tools that allow us to

(1) Determine the critical path—the chain of tasks that determines the duration of the project,

(2) Establish“most likely” time estimates for individual tasks by applying statistical models, and

(3) Calculate “boundary times” that define a time “window” for a particular task.

Gantt Chart

When creating a software project schedule, you begin with a set of tasks (the work breakdown structure). If automated tools are used, the work breakdown is input as A task network or task outline. Effort, duration, and start date are then input for each task. In addition, tasks may be assigned to specific individuals.
As a consequence of this input, a time-line chart, also called a Gantt chart, is generated. A time-line chart can be developed for the entire project. Alternatively, separate charts can be developed for each project function or for each individual working on the project.

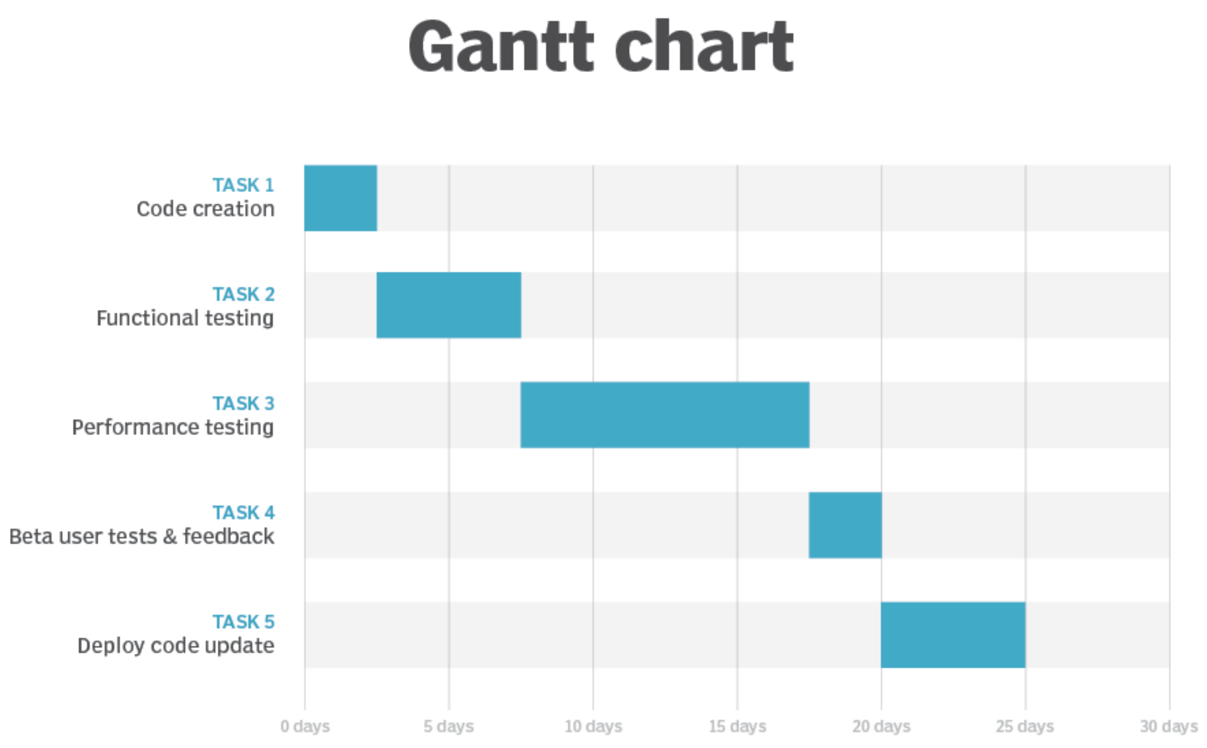
Figure a illustrates the format of a time-line chart. It depicts a part of a software project schedule that emphasizes the concept scoping task for a word-processing (WP) software product. All project tasks (for concept scoping) are listed in the left hand column. The horizontal bars indicate the duration of each task. When multiple bars occur at the same time on the calendar, task concurrency is implied. The diamonds indicate milestones.

Scheduling

3.1 Schedule table

WORK TASKS	**SCHEDULE**
1. IDENTIFY NEEDS AND BENEFITS	
Meet with concerned members	Week 1
Identify needs and project constraints	Week 1
Establish problem statement	Week 2
Milestone	Week 2
2. REQUIREMENT ANALYSIS	
Detailed discussion of the project	Week 3
Creating Data flow Diagram	Week 4
Data Dictionary	Week 5
Milestone	Week 5
3. PROJECT MANAGEMENT	
Computing F.P. and Effort	Week 5
Schedule table	Week 6
Risk table	Week 7
Timeline Chart	Week 8
Milestone	Week 8
4. DESIGN ENGINEERING	
Architectural Design	Week 8
Data Design	Week 9
Pseudo Code	Week 9
Milestone	Week 10
5. TESTING	Week 12

GANTT CHART FOR PROJECT



Architectural design

Requirements of the software should be transformed into an architecture that describes the software's top-level structure and identifies its components. This is accomplished through architectural design (also called **system design),** which acts as a preliminary 'blueprint' from which software can be developed. **IEEE** defines architectural design as 'the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.' This framework is established by examining the software requirements document and designing a model for providing implementation details. These details are used to specify the components of the system along with their inputs, outputs, functions, and the interaction between them. An architectural design performs the following functions.

1.It defines an abstraction level at which the designers can specify the functional and performance behaviour of the system.

2.It acts as a guideline for enhancing the system (when ever required) by describing those features of the system that can be modified easily without affecting the system integrity.

3.It evaluates all top-level designs.

4.It develops and documents top-level design for the external and internal interfaces. 5.It develops preliminary versions of user documentation.

6.It defines and documents preliminary test requirements and the schedule for software integration.

Though the architectural design is the responsibility of developers, some other people like user representatives, systems engineers, hardware engineers, and operations personnel are also involved. All these stakeholders must also be consulted while reviewing the architectural design in order to minimize the risks and errors.

Architectural Design Representation

Architectural design can be represented using the following models.

Structural model: Illustrates architecture as an ordered collection of program components

Dynamic model: Specifies the behavioral aspect of the software architecture and indicates how the structure or system configuration changes as the function changes due to change in the external environment

1. **Process model:** Focuses on the design of the business or technical process, which must be implemented in the system
2. **Functional model:** Represents the functional hierarchy of a system
3. **Framework model:** Attempts to identify repeatable architectural design patterns encountered in similar types of application. This leads to an increase in the level of abstraction.

RISK ANALYSIS

Identifying potential risks and developing a plan to mitigate, monitor and manage risks is of paramount importance. Risk analysis enables us to build a risk table by providing detailed guidelines in identification and analysis of risk. Points to be considered are

- Risk avoidance
- Risk monitoring
- Risk management and contingency plan.For all activities that lie above the cut-off point, a mitigation plan has been developed to mitigate the risk. A plan of action is structured, and the risk is being monitored at all the phases, i.e. a number of factors are considered.

In our context the risks for which a mitigation plan has been put into place are:

1. Lack of clear product vision.
2. No. of people may be inadequate to do the job.
3. Delivery date may extend.
4. Lack of documentation.
5. Users may change the requirements.
6. End users may resist the system.
7. Chances of Data Redundancy.
8. If authorisation is not maintained properly then it causes data tampering.
9. Project has limited features might not be profitable

Impact Values for RISK TABLE
1- Catastrophic
2-Critical
3-Marginal
4-Negligible

RISK ITEMS THEIR CATEGORIES,IMPACTS AND THEIR MITIGATION PLAN

RISK	CATEGORY	PROBABILITY(P)	IMPACT(I)	EXPOSURE E=P*I	RMMMPLAN
SOME TEAM MEMBERS LEAVE THE PROJECT IN BETWEEN	TECHNICAL RISK	20%	2	0.4	USE BACKUP STAFFS WHO KNOWS WHAT WAS GOING ON PROJECT
DELIVERY DEADLINE TIGHTENED	PROJECT RISK	30%	1	0.3	TEAM MAY USE EXTRA MEMBERS TO DO JOB ON SCHEDULED TIME
LOSING OF ALL PROJECT DATA THIS MAY HAPPEN DUE TO HARD DISK FAILURE	PROJECT RISK	20%	2	O.2	CARRY OUT BACKUP OF ESSENTIAL DATABASES,SOURCE CODE ETC.
TEAM DISTENSION /LACK OF COHESION	PROJECT RISK	10%	3	0.3	WE MAKE SOME RULES ON HOW WE CONSULT EACH OTHER

TESTING

TESTING STRATEGY

UNIT TESTING

- Focuses testing on the function or software module.
- Concentrates on the internal processing logic and data structures.
- Is simplified when a module is designed with high cohesion.
- Reduces the number of test cases.

- Allows errors to be more easily predicted and uncovered.
- Concentrates on critical modules and those with high cyclomatic complexity when testing resources are limited.

Targets for Unit Test

- Module interface

Ensure that information flows properly into and out of the module.

- Local data structures

Ensure that data stored temporarily maintains its integrity during all steps in an algorithm execution

- Boundary conditions

Ensure that the module operates properly at boundary values established to limit or restrict processing

- Independent paths (basis paths)

Paths are exercised to ensure that all statements in a module have been executed at least once

Common Computational Errors in ExecutionPaths

- Misunderstood or incorrect arithmetic precedence
- Mixed mode operations (e.g., int, float, char)
- Incorrect initialization of values
- Precision inaccuracy and round-off errors Incorrect symbolic representation of an expression (int vs. float)

WHITE BOX TESTING

White Box Testing is testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose.
is used to test areas that cannot be reached from a black box level.
Using white box testing we can derive test cases that:

- **Guarantee that all independent paths within a module have been exercised at least once.**
- **Exercise all logical decisions on their true and false sides.**

• **Execute all loops at their boundaries and within their operational bounds.**
• **Execute internal data structure to assure their validity.**

Black Box Testing-

Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated as a black box .you cannot see into it. The test provides inputs and responds to outputs without considering how the software works. It uncovers a different class of errors in the following categories:

• **Incorrect or missing function.**
• **Interface errors.**
• **Performance errors.**
• **Initialization and termination errors.**
• **Errors in objects.**

Advantages:

• The test is unbiased as the designer and the tester are independent of each other.
• The tester does not need knowledge of any specific programming languages.
• The test is done from the point of view of the user, not the designer.
• Test cases can be designed as soon as the specifications are complete.

<u>Conclusion</u>

The entire project has been developed and deployed as per the requirements stated by the user. It is found to be bug free as per the testing standards that are implemented.

The whole system's activities are divided into two major parts like User and admin. For implementing the system Django framework is used. The system comprise of following features:

- Login Super Admin,admin and user.
- Mail option for user
- Record requests from the user
- Update details

There are also few features which can be integrated with this system to make it more flexible. Below list shows the future points to be consider :

- Getting the current status of the user.
- Including a different module for colleges

Finally, we like to conclude that we put all our efforts throughout the development of our project and tried to fulfill most of the requirements of the user.

REFERENCES

1.IEEE Software Engineering Standards Committee, "IEEE Std 830-1998, IEEE Recommended Practice forSoftware Requirements Specifications", October 20, 1998.
[IEEE] The applicable IEEE standards are published in "IEEE Standards Collection," 2001 edition.

2.The principal source of text book material is "Software Engineering PRACTITIONER'S APPROACH" by Roger S. Pressman.

3.The principal source of text book material is “A Concise Introduction to Software Engineering” by Pankaj Jalote.

4. https://www.python.org

5. *https://docs.djangoproject.com/en/4.2*