AUTOMATED LEAVE MANAGEMENT SYSTEM

A project report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

By

NADHA ASHRAF	**(REG NO. DVATSCS030)**
PRANOY K.C.	**(REG NO. DVATSCS031)**
ABIN SHELDON JIMMINGTON	**(REG NO. DVATSCS002)**

Under the Guidance of

MRS. ANJANA T.K.

DEPARTMENT OF COMPUTER SCIENCE

ST. JOSEPH'S COLLEGE (AUTONOMOUS), DEVAGIRI, CALICUT

APRIL 2022

PROJECT REPORT ON

AUTOMATED LEAVE MANAGEMENT SYSTEM

CERTIFICATE

This is to certify that the project report titled

AUTOMATED LEAVE MANAGEMENT SYSTEM

Submitted by

NADHA ASHRAF	**(REG NO. DVATSCS030)**
PRANOY K.C.	**(REG NO. DVATSCS031)**
ABIN SHELDON JIMMINGTON	**(REG NO. DVATSCS002)**

St. Joseph's College, Devagiri in partial fulfilment of the requirements for the award of degree of Bachelor of Science in Computer Science, during the year 2019-2022.

Internal Guide Head of Department

Internal Examiner External Examiner

DECLARATION

We hereby declare that the project entitled “Automated Leave Management System” has been undertaken by us for the award of the degree of Bachelor of Science in Computer Science. We have completed this project under the guidance of Mrs. Anjana T.K. Department of Computer Science, St. Joseph’s College (Autonomous), Devagiri, Calicut.

Place: Calicut
Date:

NADHA ASHRAF	(REG NO. DVATSCS030)
PRANOY K.C.	(REG NO. DVATSCS031)
ABIN SHELDON JIMMINGTON	(REG NO. DVATSCS002)

ST. JOSEPH'S COLLEGE (AUTONOMOUS)
PRO DEO ET PATRIA
DEVAGIRI, CALICUT

CERTIFICATE

This is to certify that the project report titled "**Automated Leave Management System"** a project work done by them during the academic year 2020-2021 under our guidance and supervision in partial fulfilment of the requirement of Bachelor of Science in Computer Science.

Place : Devagiri

Date : (Signature)

Mrs. Anjana T.K.

(Signature)

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to Mr. Sabu K Thomas, Principal of St. Joseph’s College (Autonomous), Devagiri, Calicut for providing necessary facilities for developing our project.

We are also grateful to Mrs. Asha Unnikrishnan, Head of the Department in charge, as well as Mrs. Anjana T.K., our internal guide for their immense support and valuable suggestions. We are also grateful to Mr. T. Likhil, managing director, RISS Technologies, Calicut for providing assistance to do this work in his organization.

NADHA ASHRAF	(REG NO. DVATSCS030)
PRANOY K.C.	(REG NO. DVATSCS031)
ABIN SHELDON JIMMINGTON	(REG NO. DVATSCS002)

SYNOPSIS

An online leave management system is a web-based leave management application that automates every step of the employee leave management process without compromising on functionality. Unlike on premise leave management tools, an online leave management system gives employees the freedom to apply, approve, reject, and manage leave requests from any place, any time, and from any device. Our proposed system is an automated leave management system that computerizes the process of applying for and approving a leave of absence and making necessary changes to the schedule accordingly. The system stores the timetables for all the classes beforehand. A faculty member applies for a leave and once leave is approved, the system automatically adjusts the timetable to accommodate the absence of that faculty member by assigning it to another member who is available at that time.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER- I

INTRODUCTION

1.1 LEAVE MANAGEMENT SYSTEM-AN OVERVIEW

Our Automated Leave Management System is a web-based platform for staff members of educational institutions to manage leave applications, timetables and absences of staff members. The system automatically accommodates absences of staff members by assigning substitution to other unoccupied staff members automatically. The staff member then gets notified on this change in their timetable via email.

1.2 INTRODUCTION & MISSION OF PROJECT

Manual leave applications and timetable assignment can prove to be a laborious task for teachers who otherwise, 2have lots of other work to complete on a daily basis. Our Leave Management System simply aims to take this task of applying for leave and finding suitable substitution off of their plates. This system makes applying for a leave as easy as clicking a few buttons.

Further, it automatically assigns substitution to unoccupied staff members so the HOD does not have to do this manually.

1.3 BACKGROUND STUDY

1.3.1 THE ORGANIZATION PROFILE

RISS technologies, an ISO 9001:2008 certified company focuses on transforming and running business processes and operations including those that are complex and industry-specific. Our loom is distinctive: through an unbiased, agile combination of smarter process science, targeted technology and advanced analytics, we help our clients become more competitive by making their enterprises more intelligent: adaptive, innovative, globally effective, and connected to their own clients.

We provide end-to-end expert IT solutions across entire software delivery cycle. RISS cater to every single need of our clients and help them in accomplishing their goals and objectives. RISS has significant expertise accumulated over these 15 years of specialized work with hundreds of enterprises, and we remain loyal to our heritage of operational excellence as an extension of our clients' business reflected by the best client satisfaction scores in the industry**.**

1.3.2 STUDY ON EXISTING SYSTEM

The existing system of leave management requires the employees to apply for leave manually by personally contacting the heads of their departments. The HOD then approves the leave application and manually alters the timetable to accommodate the absence of that teacher by distributing his or her classes to other available faculty members.

1.3.3 DISADVANTAGES OF EXISTING SYSTEM

- Requires more man power.
- Time consuming.
- Consumes large volume of paper work.
- Needs manual calculations.
- Larger scope for manual errors.
- To avoid all these limitations and make the working more accurate, the system needs to be computerized in a better way.

CHAPTER- II

SYSTEM ANALYSIS

2 SYSTEM ANALYSIS

A system study is a process of studying a procedure to identify the objectives and purposes of a system as well as to analyse the existing system's problems and drawbacks. The system analysis will create systems and procedures that will achieve these objectives as well as solutions to the problems in an efficient way. It is the process of gathering and interpreting facts, diagnosing problems and using the information to recommended improvements to the system. Training, experience and common sense are required for the collection of the information needed to do the analysis.

2.1 STUDY ON PROPOSED SYSTEM

Our proposed system is an automated leave management system that can be run using any web browser. This system computerizes the process of applying for and approving a leave of absence and making necessary changes to the schedule accordingly.

The system stores the timetables for all the classes beforehand. A faculty member applies for a leave and once leave is approved, the system automatically adjusts the timetable to accommodate the absence of that faculty member by assigning it to another member who is available at that time.

This avoids the hassle of manually examining the timetables and assigning substitute teachers based on availability. This system thus ensures that no class is left unmonitored during class hours.

2.1.1 ADVANTAGES OF PROPOSED SYSTEM

- More efficient.
- Less time consuming.
- This web application can be accessed at anytime and anywhere.
- Provides mobility.
- Since the timetables automatically alters itself, it requires less manual labour.
- Less scope for human error.

2.2 USER REQUIREMENT SPECIFICATION

After thorough analysis, our system has been presented with the following modules:

1. Admin
2. Staff

ADMIN

The overall working of the website is controlled by the admin. Functionalities performed by the admin are as follows:

- Login
- Add and manage course
- Add and Manage subjects
- Add and manage staff details
- Allocate staff to each subject
- Create and manage timetable
- Approve and reject leave requests
- View and Reply to complaints sent by staff

STAFF

Staff module can perform the following functions:

- Login
- View Profile
- Change Password
- View allocated Subjects
- View timetable
- Request for Leave
- View Leave request status
- Send Complaints
- View complaint replies
- Get notified on changes made in the timetable

2.3 SOFTWARE REQUIREMENTS

Once the system requirements are found then we have to determine whether a particular software package fits for those system requirements.

FRONT END:

HTML

Hypertext mark up language is a standard mark-up language for creating web pages. It describes the structure of a Web page and consists of a series of elements which tell the browser how to display the contents. It is used by many programmers to create a website and is easy and simple to learn.

PROGRAMMING LANGUAGES

PYTHON

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by Meta programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Python is easy to learn and is platform independent. It has extensive libraries and support POP and OOP. So we preferred Python language as our back end for web.

BACK END:

MySQL

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing. It is free of cost and has high performance. It is very fast and have memory efficiency database. As it is simple to use we choose MySQL as the database at the back end.

2.4 FEASIBILITY STUDY

A feasibility study is an assessment of the practicality of a proposed system or project. A feasibility study aims to objectively and rationally uncover weakness and strengths of an existing business or proposed venture, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects of success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained. A feasibility study evaluates the project's potential for success; therefore, perceived objectively is an important factor in the credibility of the study for potential investors and lending institutions. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based. The feasibility study is done in these phases.

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

TECHNICAL FEASIBILITY

It investigates the technical feasibility of each implementation alternative. It analyses and determines whether the solution can be supported by existing technology or not. The analyst determines whether current technical resources be upgraded or added it that fulfil the new requirements. The analyst must find out whether current technical resources can be upgraded or added in a manner that fulfil the request under considerations. Our software is more user friendly. We are using python language for coding. So, it is easy to understand and it is more readable.

ECONOMICAL FEASIBILITY

The purpose of an economic feasibility study is to demonstrate the net benefit of a proposed project for accepting or disbursing electronic funds/benefits and costs to the agency, other state agencies and the general public as a whole. It is evaluating the effectiveness of candidate system by using cost/benefit analysis method. It demonstrates the net benefits from the candidate system in terms of benefits and costs to the organization. Software is said to be economically feasible if it focuses on issues like cost incurred on software development to produce long term gains for an organization, cost required to conduct full software investigation, cost of hardware, software, development team, and training. Our application can be economically feasible since it can be used by all sections of the society.

OPERATIONAL FEASIBILITY

It determines whether the system is operating effectively once it is developed and implemented. It ensures that the management should support the proposed system and its working feasible in the current organizational environment. It analyses whether the users will be affected and they accept the modified or new business methods that affect the possible system benefits. It also ensures that the computer resources and network architecture of candidate system are workable. It defines the urgency of the problem and the acceptability of any solution; if the system is developed, will it be used? Includes people oriented and social issues, internal issues, such as manpower problems, labour objections, manager resistance, organizational conflicts and policies; also, external issues, including legal aspects and governments regulations, also social acceptability of the new system. Our Leave management System can be successful in terms of operational feasibility because it can be used on a daily basis for leave management purposes.

2.5 SYSTEM SPECIFICATION

2.5.1 HARDWARE SPECIFICATION

- Hardware: 64 bit Processor
- RAM: Minimum 3GB
- Hard Disk: 20 GB
- Key Board: Standard Windows Keyboard
- Mouse: Two or Three Button Mouse

● Monitor: SVGA

2.5.2 SOFTWARE SPECIFICATION

● Operating System: Windows 8 or above

● Technology: python //HTML/CSS

● IDE: PyCharm,

● Framework: Flask

● Database: MySQL

2.6.1 COST ESTIMATION

● H/W:40000

● S/W:5000

● PERSONNEL:3000

● OTHER CHARGES:2000

2.6.2 SCHEDULING

■ Requirements collection- 2weeks

■ Analysis and design- 2weeks

■ Coding- 4 weeks

■ Testing- 1week

CHAPTER-III

DESIGN AND DEVELOPMENT PROCESS

3.1 FUNDAMENTAL DESIGN CONCEPTS

A set of fundamental software design concepts have developed over the history of software engineering. Each provides the software designer with a foundation from which more sophisticated design methods can be applied. Some important software design concepts that span both traditional and object-oriented software development:

1. **Abstraction**

   Abstraction simply means to hide the details to reduce complexity and increases efficiency or quality. Different levels of abstraction are necessary and must be applied at each stage of the design process so that any error that is present can be removed to increase the efficiency of the software solution and to refine the software solution.

2. **Architecture**

   Architecture simply means a technique to design a structure of something. Architecture in designing software is a concept that focuses on various elements and the data of the structure. These components interact with each other and use the data of the structure in architecture.

3. **Separation of concerns**

   Separation of concerns is a design concept that suggests any complex problem can be more easily handled if it is subdivided into pieces that can each be solved and/or optimized independently. By separating concerns into smaller, and therefore more manageable pieces, a problem takes less effort and time to solve.

4. **Modularity**

   Modularity simply means to divide the system or project into smaller parts to reduce the complexity of the system or project. In the same way, modularity in design means to subdivide a system into smaller parts so that these parts can be created independently and then use these parts in different systems to perform different functions.

5. **Information Hiding**

   Information hiding simply means to hide the information so that it cannot be accessed by an unwanted party. In software design, information hiding is achieved by designing the modules in a manner that the information gathered or contained in one module is hidden and it can’t be accessed by any other modules.

6. **Refinement**

   Refinement simply means to refine something to remove any impurities if present and increase the quality. The refinement concept of software design is actually a process of developing or presenting the software or system in a detailed manner that means to elaborate a system or software. Refinement is very necessary to find out any error if present and then to reduce it.

7. **Object-Oriented Design Concepts**

   The object-oriented (OO) paradigm is widely used in modern software engineering. OO design concepts such as classes and objects, inheritance, messages, and polymorphism, among others are the different features of OO paradigm.

3.2 DESIGN NOTATIONS

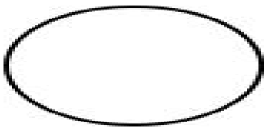
3.2.1 DATA FLOW DIAGRAM

A graphical representation is used to describe and analyse the movement of data through a system manual or automated including the processes, storing of data and delays in the system. Data flow diagrams are the central tool and the basis from which other components are developed. The transformation of the data from input to output through process may be described logically and independently of the physical components associated with the system. They are termed logical data flow diagrams, showing the actual implementation and the movement of data between people, departments and workstations. DFD is one of the most important modelling tools used in physical design. DFD shows the flow of data through different process in the system.

Notations used in DFD

In DFD, there are four symbols, they as follows:

- Process



- Data Store

- External Entity

- Direction of flow

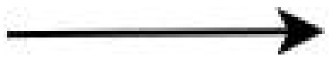


Figure 3.2.1.1 : LEVEL 0 DFD

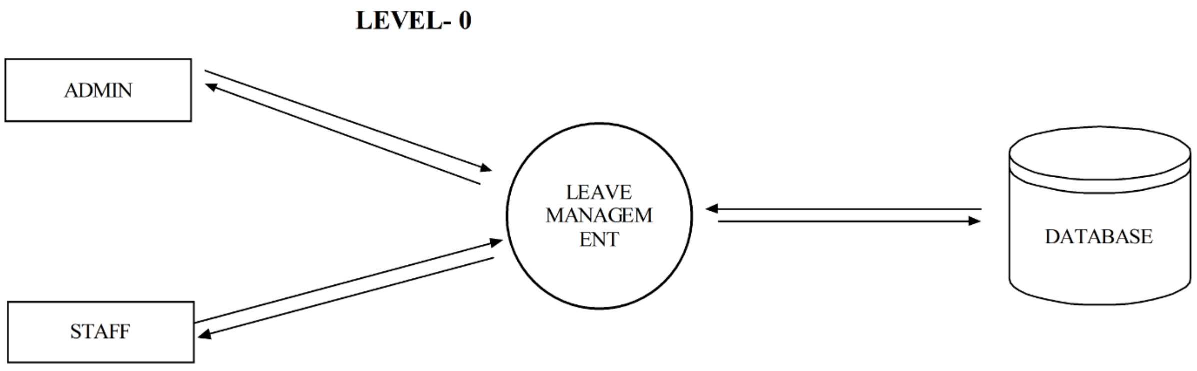


Figure 3.2.1.2 : LEVEL 1.0 DFD (Admin)

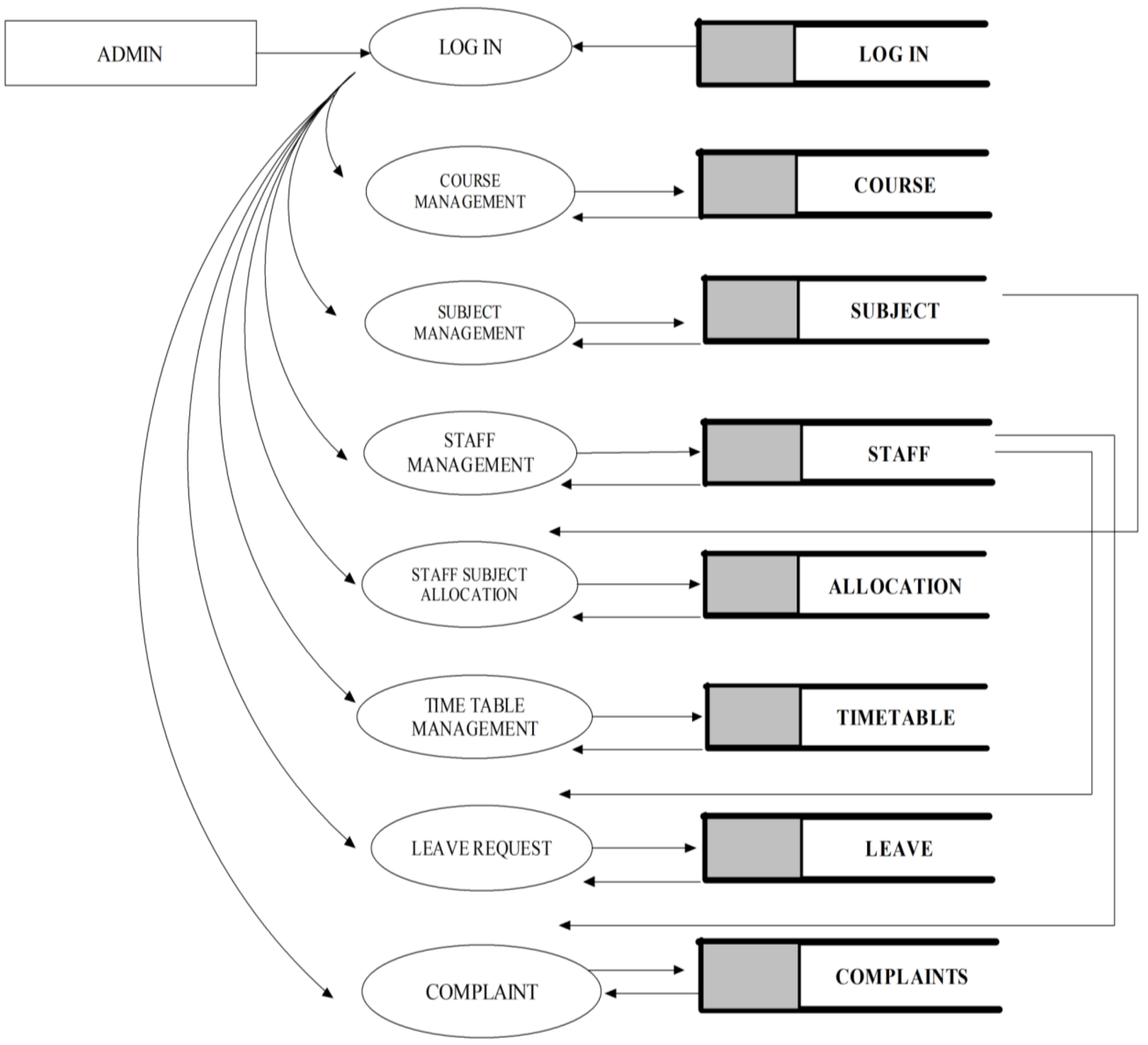
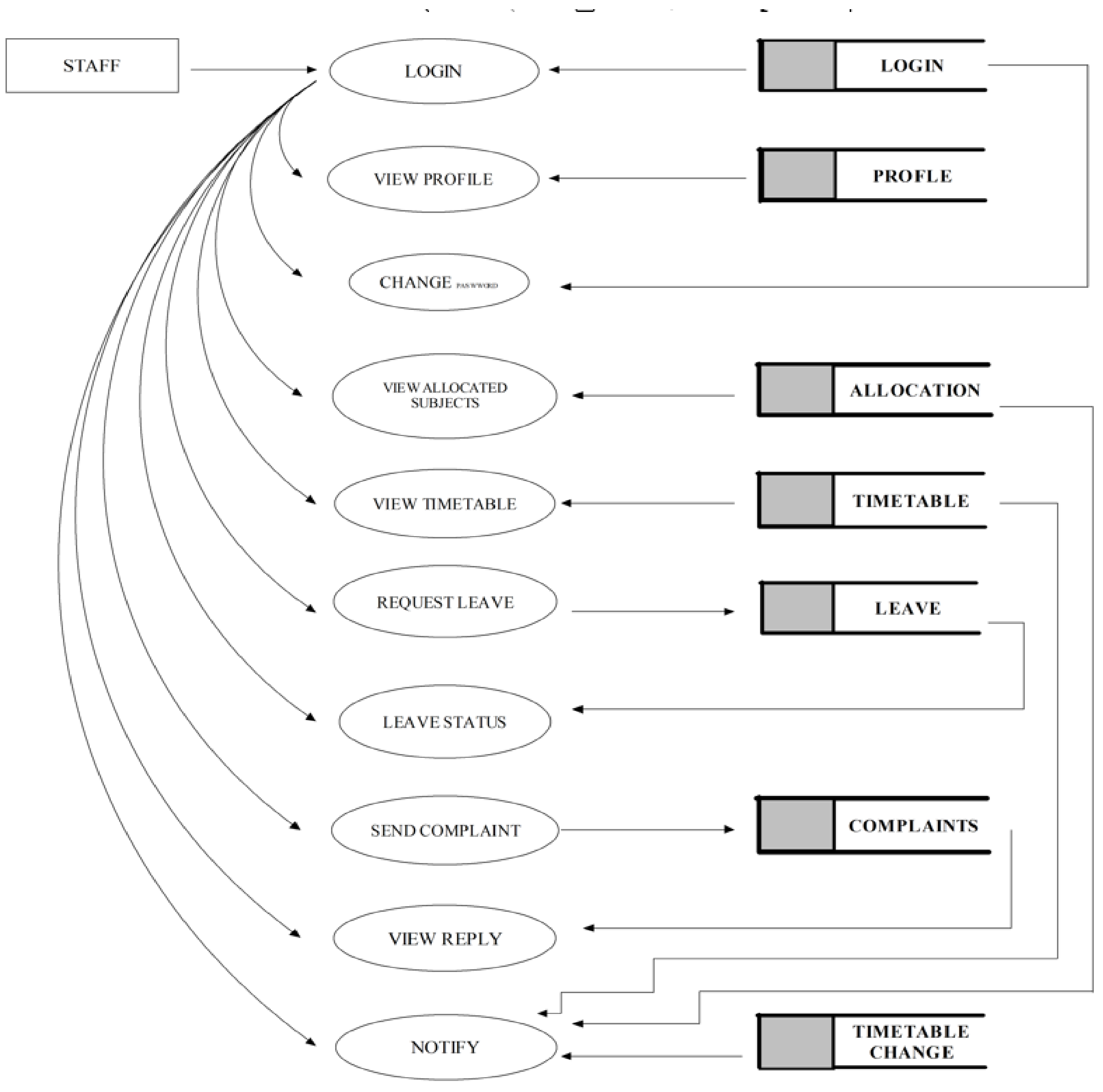


Figure 3.2.1.3 : LEVEL 1.1 DFD (User)

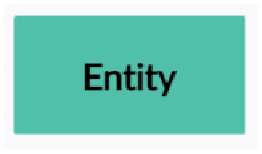


ER DIAGRAM

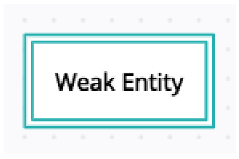
ER-Diagram is a pictorial representation of data that describes how data is communicated and related to each other. Any object, such as entities, attributes of an entity, sets of relationship and other attributes of relationship can be characterized with the help of the ER diagram.ER model is a high level conceptual data model. It represents real world entities and relationship between them. It helps to analyse data requirements systematically to produce a well designed database.

Notations used in ER Diagram

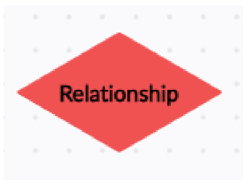
- Entity



- Weak Entity



- Relationship



- Attribute

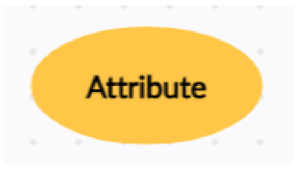
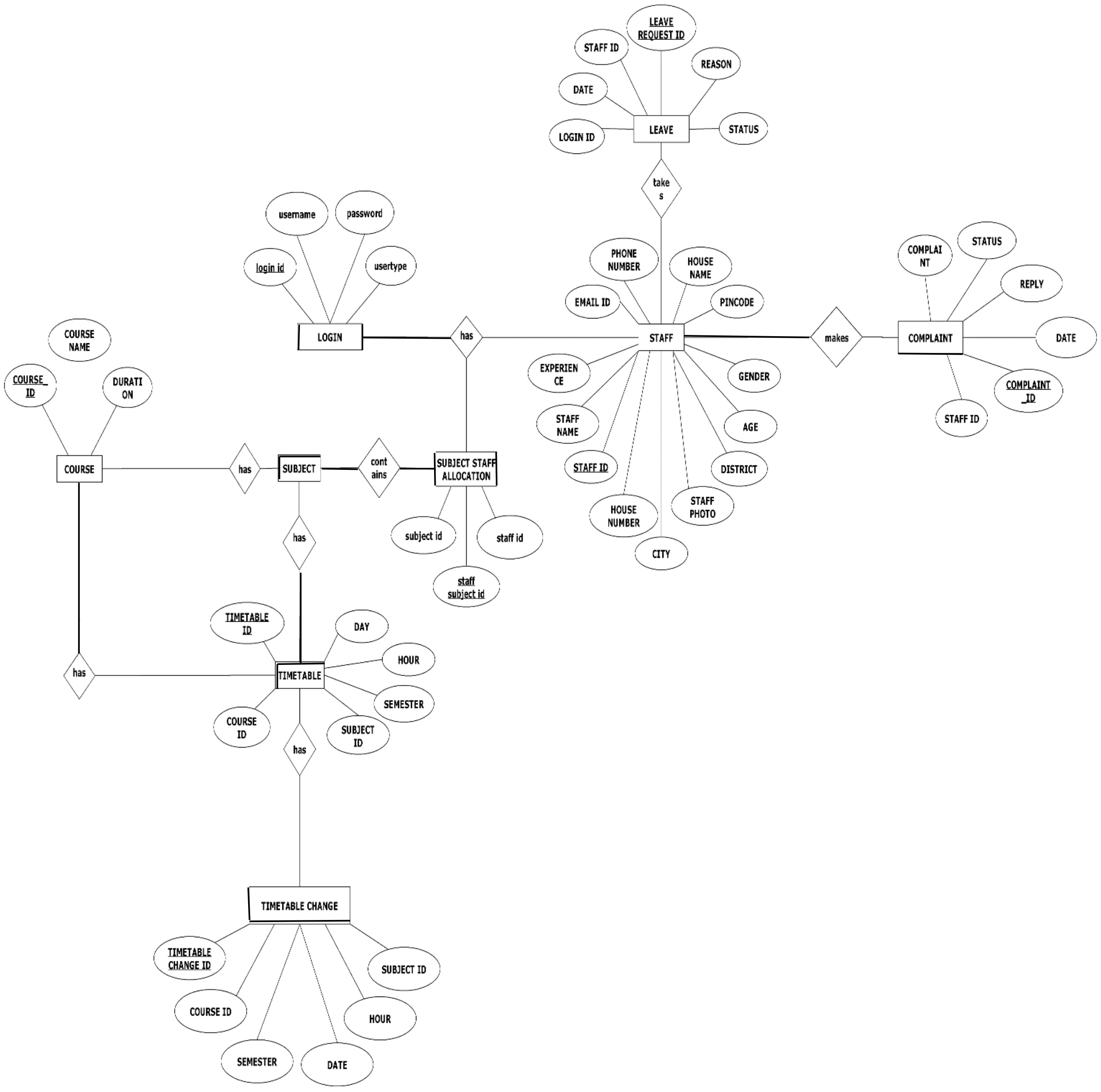


Figure 3.2.2.1 ER DIAGRAM



3.3 DESIGN PROCESS

3.3.1 DATABASE DESIGN

A table is data structure that organizes information into rows and columns. It can be used to both store and display data in a structured format. Database often contain multiple tables, with each one designed for specific purpose. Each table may include its own set of fields, based on what data the table need to store. In database tables, each field is considered a column, while each entry (or record) is considered a row. A specific value can be accessed from the table by requesting data from an individual column and rows. There are primary key fields for uniquely identifying a record in a table.

Table 3.3.1.1 : LOGIN

SL NO	FIELD NAME	DATATYPE	CONSTRAINTS
1.	Login ID	varchar	Primary key
2.	Username	varchar	Not NULL
3.	Password	varchar	Not NULL
4.	Usertype	varchar	Not NULL

Table 3.3.1.2: COURSE

SL NO	FIELD NAME	DATATYPE	CONSTRAINTS
1.	Course ID	varchar	Primary Key
2.	Course Name	varchar	Not NULL
3.	Duration	int	Not NULL

Table 3.3.1.3 SUBJECT

SL NO	FIELD NAME	DATATYPE	CONSTRAINTS
1.	Subject ID	varchar	Primary key
2.	Subject Name	varchar	Not NULL
3.	Course ID	varchar	Foreign Key
4.	Semester	varchar	Not NULL

Table 3.3.1.4 : STAFF

SL NO	FIELD NAME	DATATYPE	CONSTRAINTS
1.	Staff ID	varchar	Primary Key
2.	Staff Name	varchar	Not NULL
3.	Experience	int	Not NULL
4.	Staff Photo	file	Not NULL
5.	Age	int	Not NULL
6.	Gender	varchar	Not NULL
7.	Mail ID	varchar	Not NULL
8.	Phone Number	int	Not NULL
9.	Login ID	varchar	Foreign Key
10.	House Name	Varchar	Not NULL
11.	House Number	Varchar	Not NULL
12.	City	Varchar	Not NULL
13.	Pin code	int	Not NULL

Table 3.3.1.5 : SUBJECT STAFF ALLOCATION

SL NO	FIELD NAME	DATATYPE	CONSTRAINTS
1.	Staff Subject ID	varchar	Primary key
2.	Subject ID	varchar	Foreign key
3.	Subject Name	varchar	Foreign key

Table 3.3.1.6 TIMETABLE

SL NO	FIELD NAME	DATATYPE	CONSTRAINTS
1.	Timetable ID	varchar	Primary key
2.	Day	Varchar	Not NULL
3.	Hour	Varchar	Not NULL
4.	Course ID	Varchar	Foreign Key
5.	Semester	varchar	Not NULL
6.	Subject ID	varchar	Foreign Key

Table 3.3.1.7: LEAVE

SL NO	FIELD NAME	DATATYPE	CONSTRAINTS
1.	Leave Request ID	varchar	Primary Key
2.	Staff ID	varchar	Foreign Key
3.	Date	date	Not NULL
4.	Reason	varchar	Not NULL
5.	Status	varchar	Not NULL

Table 3.3.1.8 : COMPLAINT

SL NO	FIELD NAME	DATATYPE	CONSTRAINTS
1.	Complaint ID	varchar	Primary Key
2.	Staff ID	varchar	Foreign Key
3.	Date	date	Not NULL
4.	Complaint	Varchar	Not NULL
5.	Reply	Varchar	Not NULL
6.	Status	varchar	Not NULL

3.3.2 NORMALIZATION

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. In our design the tables have been normalized up to third normalization form. Normalization applied during database design are:

- First Normal Form (1NF)
- Second Normal Form (2 NF)
- Third Normal Form (3 NF)

First Normal Form

If a relation contains composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute. Our application satisfies 1 NF. The first normal form states that:

- Every column in the table must be unique
- Separate tables must be created for each set of related data

- Each table must be identified with a unique column or concatenated columns called the primary key
- No rows may be duplicated
- No columns may be duplicated
- No row/column intersections contain a null value

Second Normal Form

A relation is said to be in 2NF if and only if it satisfies all 1NF conditions for the primary key and every non-primary key attribute of the relation is fully dependent on its primary key alone. If a non-key attribute is not dependent on the key, it should be removed from the relation and places as a separate relation. i.e., the fields of the table in 2NF are all related to primary key.

Third Normal Form

A relation is said to be in 3NF if and only if it is in 2NF and more over non-key attribute of the relation should not depend on other non-key attributes. The data in the system has to be stored and retrieved from database. Designing the database is a part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system.

3.3.3 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or print the document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

3.3.4 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

CHAPTER -IV

TESTING AND IMPLEMENTATION

4.1 TESTING

Software testing is the process of verifying a system with the purpose of identifying any errors, gaps or missing requirement versus the actual requirement. Software testing is broadly categorized into two types - functional testing and non-functional testing. When to start test activities: Testing should be started as early as possible to reduce the cost and time to rework and produce software that is bug-free so that it can be delivered to the client. However, in Software Development Life Cycle (SDLC), testing can be started from the Requirements Gathering phase and continued till the software is out there in productions. It also depends on the development model that is being used. For example, in the Waterfall model, testing starts from the testing phase which is quite below in the tree, but in the V-model, testing is performed parallel to the development phase.

4.1.1 TESTING METHODOLOGIES

MANUAL TESTING

Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test cases are planned and implemented to complete almost 100 percent of the software application. Test case reports are also generated manually.Manual Testing is one of the most fundamental testing processes as it can find both visible and hidden defects of the software. The difference between expected output and output, given by the software, is defined as a defect. The developer fixed the defects and handed it to the tester for retesting.

Manual testing is mandatory for every newly developed software before automated testing. This testing requires great efforts and time, but it gives the surety of bug-free software. Manual Testing requires knowledge of manual testing techniques but not of any automated testing tool. If the test engineer does manual testing, he/she can test the application as an end-user perspective and get more familiar with the product, which helps them to write the correct test cases of the application and give the quick feedback of the application.

AUTOMATION TESTING

When the testing case suites are performed by using automated testing tools is known as Automation Testing. The testing process is done by using special automation tools to control the execution of test cases and compare the actual result with the expected result. Automation testing requires a pretty huge investment of resources and money.

Generally, repetitive actions are tested in automated testing such as regression tests. The testing tools used in automation testing are used not only for regression testing but also for automated GUI interaction, data set up generation, defect logging, and product installation. The goal of automation testing is to reduce manual test cases but not to eliminate any of them. Test suits can be recorded by using the automation tools, and tester can play these suits again as per the requirement. Automated testing suites do not require any human intervention.

4.1.2 DIFFERENT TESTING

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing
- Validation Testing

UNIT TESTING

The first level of testing is called as unit testing. Here the different modules are tested and the specification produced during design for the modules. Unit testing is essential for verification of the goal and to test the internal logic of the modules. Unit testing is conducted to different modules of the project. Errors were noted down and corrected down immediately and the program clarity was increased. The testing was carried out during the programming stage itself. In this step each module is found to be working satisfactory as regard to be expected out from the module. For example, the Login page is tested against three different states that are a positive input, a negative input and a 0 input. Testing with a positive input, and with a negative input will behave as expected. Testing with a 0 input however will yield a surprise. This is just one example of why it makes sense to focus on testing the different states of your code.

INTEGRATION TESTING

The second level of testing includes integration testing. It is a systematic testing of constructing structure. At the same time tests are conducted to uncover errors with the interface. It need not to be the case, that software whose modules when run individually showing results will also show perfect results when run as a whole. The individual modules are tested again and the results are verified. The goal is to see if the modules integrated between the modules. This testing activity can be considered as testing the design and emphasizes on testing modules interaction.

- Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

- Bottom Up Integration

This method begins the construction and testing with the modules at the lowest in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated.

SYSTEM TESTING

Testing is an activity to verify that a correct system is being built and is performed with the intent of finding faults in the system. However not restricted to being performed after the development phase is complete, but this is to carry out in parallel with all stages of system development, starting with requirements specification. Testing results, once gathered and evaluated, provide a qualitative indication of software quality and reliability and serve as a basis for design modification if required. A project is said to be incomplete without proper testing.
System testing is a process of checking whether the developed system is working according to the original objectives and requirements. The system should be tested experimentally with test

data so as to ensure that system works according to the required specification. When the system is found working, test it with actual data and check performance.

ACCEPTANCE TESTING

Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it has met the required criteria for delivery to end users.

VALIDATION TESTING

- Validation succeeds when software functions in a manner that can be responsibly expected by the customer. It covers the following:
- Validation test criteria: Performance, functional characteristics and uncovered deviation from specification.
- Configuration review: Ensures that all the elements of software configuration have been properly developed catalogued and have support for the maintenance phase of software life cycle.
- Alpha beta testing: Alpha test is conducted by developer’s site by customers. Beta test is conducted at one or more customer site by software end user.
- Modular integration testing: Modular integration testing is done to ensure that the module is working independently. The inputs as required by the module are given as required and the output is tested as per the specification.

Front-End Validation

Everything is validated on the server to prevent someone creating an alternative client to access/manipulate database. Front end is also necessary as it improves efficiency and prevents the server being accessed with inappropriate data.

Front end validation is for data entry help and contextual messages. This ensures that the user has a hassle-free data entry experience and minimizes the round-trip required for validating correctness. Front end validation validates all input in a modern application, providing the user with quick feedback a possible issue. In the UI, tourist guide does basic input validation – like checking the mandatory fields, or validity of an email address, and updating or disabling UI controls based on that.

Back-End Validation

Back-end validation is also an essential part. It has to ensure that the data coming in is needed valid. Additionally, depending on the architecture, generally re-use the middle-tier business logic amongst multiple components so need to ensure the rules that are applied are always consistent – regardless of what the front-end logic enforces.

4.1.3 TEST CHART

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

Test case for login:

Test ID	**TEST CASE**	**TEST DATA**	**EXPECTED RESULT**	**ACTUAL RESULT**	**REMARK**
1.	check admin login with valid data	Username:admin Password: admin	admin should login to the admin dashboard	login successful	Pass
2.	check admin login with invalid data	username: marshu8@gmail.com password: 256848	admin should fail to login to admin dashboard	login failed	Pass
3.	check admin login by emptying the data and login button pressed	no data	alerted that field is empty	trigger an error alert	Pass
4.	check staff login with valid data	user name: ashaunni@gmail.com password : 9126584930	member should login to the android application	login successful	Pass

5	check member login with invalid data	username: abc@gmail.com password: 111457	member should fail to login to android application	login failed	Pass
6	check member login by emptying the data and login button pressed	no data	alerted that field is empty	trigger an error alert	Pass

4.2 QUALITY ASSURANCE & POLICIES

4.2.1 GENERIC RISKS

“Risk is future uncertain events with a probability of occurrence and potential for loss “Risk identification and management are the main concerns in every software project. Effective analysis of software risks will help to effective planning and assignments of work. Risks are identified, classified and managed before the actual execution of the program.

These Risks are classified into different categories:

1) Schedule Risk: Project schedule get slip when project tasks and schedule release risks are not addressed properly. Schedule risks mainly affect a project and finally on company economy and may lead to project failure.

Schedules often slip due to the following reasons:

- Wrong time estimation
- Resources are not tracked properly. All resources like staff, systems, skills of individuals, etc.
- Failure to identify complex functionalities and time required to develop those functionalities.
- Unexpected project scope expansions.

2) Budget Risk

- Wrong budget estimation.
- Cost overruns
- Project scope expansion

3) Operational Risks: Risks of loss due to improper process implementation failed system or some external events risks. Causes of Operational Risks:

- Failure to address priority conflicts
- Failure to resolve the responsibilities
- Insufficient resources
- No proper subject training
- No resource planning
- No communication in the team.

4) Technical Risks: Technical risks generally lead to failure of functionality and performance. Causes of Technical Risks are:

- Continuous changing requirements
- No advanced technology available or the existing technology is in the initial stages.
- The product is complex to implement.
- Difficult project modules integration.

5) Programmatic Risks: These are the external risks beyond the operational limits. These are all uncertain risks are outside the control of the program. These external events can be:

- Running out of the fund.
- Market development
- Changing customer product strategy and priority
- Government rule changes.

4.3 SYSTEM IMPLEMENTATION

System implementation is the final phase i.e., putting the utility into action. Implementation is the state in the project where theoretical design turned into working system. Implementation involves the conversion of a basic application to complete replacement with a computer system. It is the process of converting to a new or revised system design into an operational one. During the design phase, the products structure, its undergoing data structures, the general algorithms and the interfaces and control/data linkages needed to support communication among the various sub structures were established. Implementation process is simply a translation of the design abstraction into the physical realization, using the language of the target architecture.

There are three types of implementation:

- Implementation of a computer system to replace a manual system.
- Implementation of a new computer system to replace an existing one.
- Implementation of a modified application to replace an existing one computer.

The common approaches for implementation are:

- Parallel Conversion

In parallel conversion the existing system and new system operate simultaneously until the project team is confident that the new system is working properly. The outputs from the old system continue to be distributed until the new system has proved satisfactorily parallel conversion is a costly method because of the amount of duplication involved.

- Direct Conversion

Under direct conversion method the old system is discontinued altogether and the new system becomes operational immediately. A greater risk is associated with direct conversion is no backup in the in the case of system fails.

- Pilot Conversion

A pilot conversion would involve the changing over of the part of the system either in parallel or directly. Use of the variation of the two main methods is possible when part of the system can be treated as a separate entity.

- User Training

After the system is implemented successfully, training of the user is one of the most important subtasks of the developer. For this purpose, user manuals are prepared and handled over to the user to operate the developed system. Thus, the users are trained to operate the developed system.

In order to put new application system into use, the following activities were taken care of:

- Preparation of user and system documentation.
- Conducting user training with demo and hands on.
- Test run for some period to ensure smooth switching over the system.

4.3.1 IMPLEMENTATON PROCEDURES

The major implementation procedures are:

- Test plans
- Training
- Conversion

The software tools used in order to create this application are:

- JetBrains PyCharm Community Edition

4.4 SYSTEM MAINTENANCE

The maintenance is an important activity in the life cycle of a software product. Maintenance includes all the activities after the installation of software that is performed to keep the system operational. The maintenance phase of a software life cycle is the time period in which a product performs useful work. Maintenance is classified into four types.

- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance
- Preventive Maintenance

CORRECTIVE MAINTENANCE

Corrective maintenance refers to changes made to repair defects in the design, coding, or implementation of the system. Corrective maintenance is often needed for repairing processing or performance failures or making changes because of previously uncorrected problems or false assumptions. Most corrective maintenance problems surface soon after the installation. When corrective maintenance problems surface, they are typically urgent and need to be resolved to curtail possible interruptions in normal business activities.

ADAPTIVE MAINTENANCE

Adaptive maintenance involves making changes to an information system to evolve its functionality or to migrate it to different operating environment. Adaptive maintenance is usually less urgent than corrective maintenance because of business and technical changes typically occur some period of time.

PERFECTIVE MAINTENANCE

Perfective maintenance involves making enhancements to improve processing performance, interface usability, or to add desired, but not necessarily required, system features. Many system professionals feel that perfective maintenance is not really the maintenance but new development.

PREVENTIVE MAINTENANCE

Preventive maintenance is the only maintenance activity which is carried out without formal maintenance request from the user. When a software company or maintenance agency realizes that the methodologies used in a program have become obsolete, it may decide to develop or modify parts of the program, which do not confirm to the current trends. Of these types, more time and money are spending on perfect than on corrective and adaptive maintenance together.

CHAPTER-V
CONCLUSION

SCOPE FOR FURTHER ENHANCEMENTS

In future this app can be upgraded with effective features like:

- Automatic timetable generation.
- Expansion of the system to include other departments of the institution as well.
- Automatic rejection of leave requests if the staff has already exceeded their allowed leave limit.
- Facility for teachers to reject substitution under valid circumstances.

BIBILOGRAPHY

Textbooks

- Elmasri R. & Navathe S. (2007). Fundamentals of database systems. Boston: Pearson/Addison Wesley.
- Sommerville I. (2016). Software engineering. Harlow: Pearson Education.

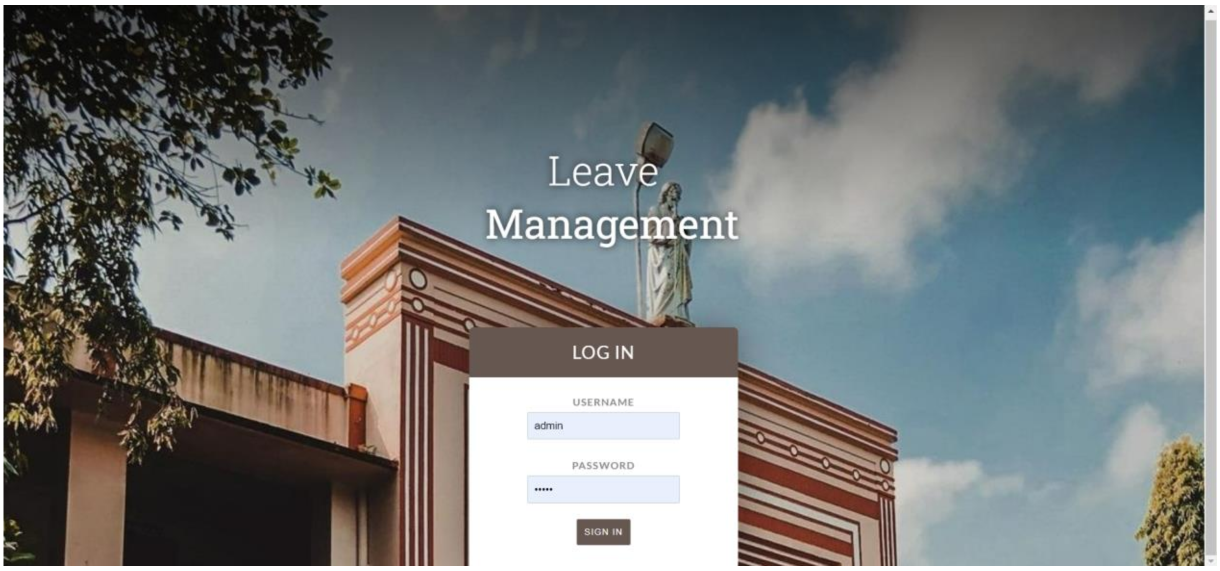
Web Sites

- www.w3schools.com
- www.udemy.com
- javatpoint

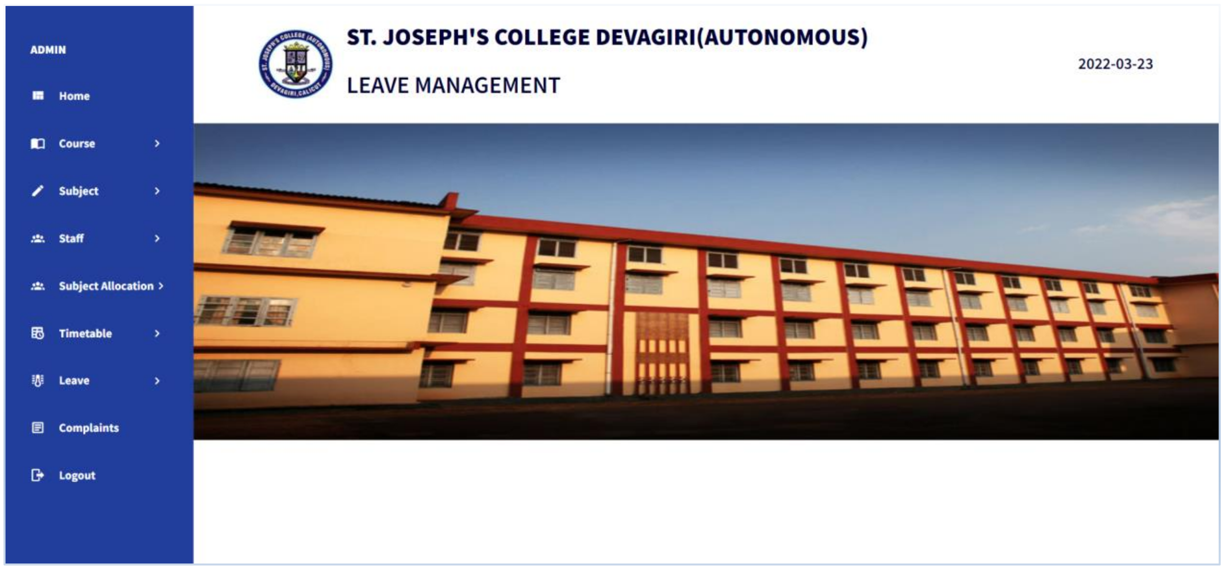
ANNEXURE-A

1.INPUT AND OUTPUT DESIGN:

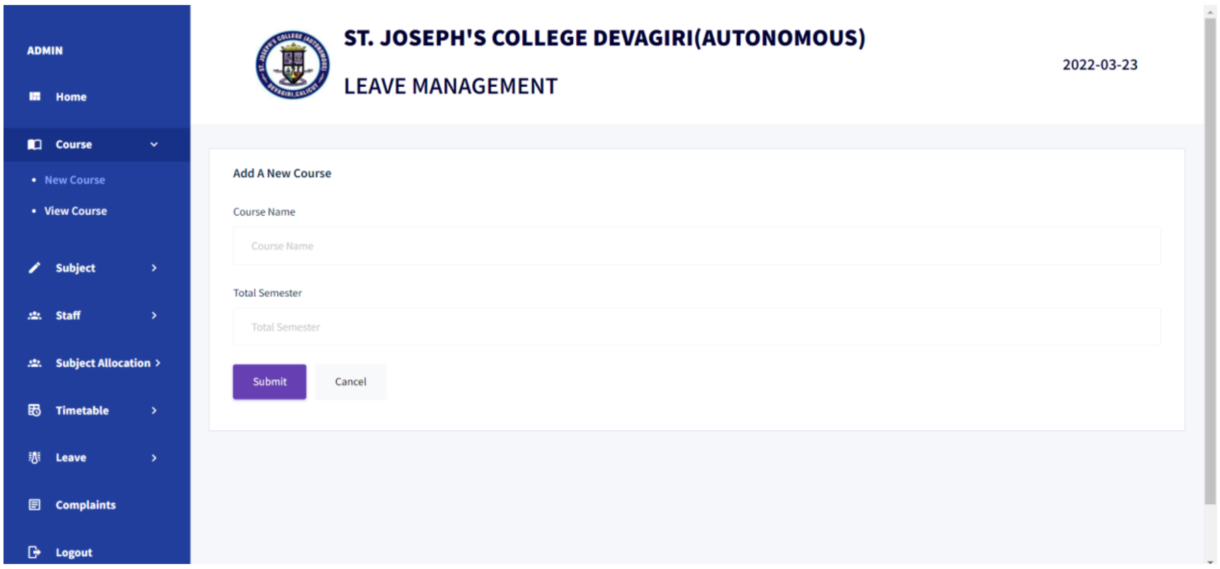
LOGIN PAGE



ADMIN HOME PAGE



ADMIN ADD COURSE



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

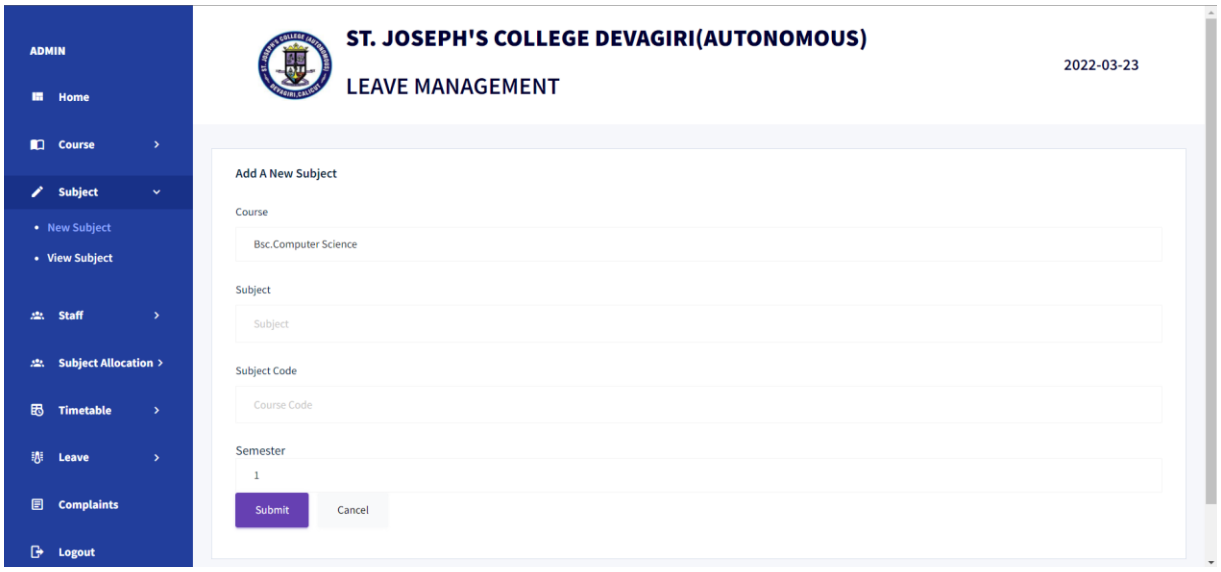
2022-03-23

Add A New Course

Course Name: Course Name

Total Semester: Total Semester

ADMIN ADD SUBJECT



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

2022-03-23

Add A New Subject

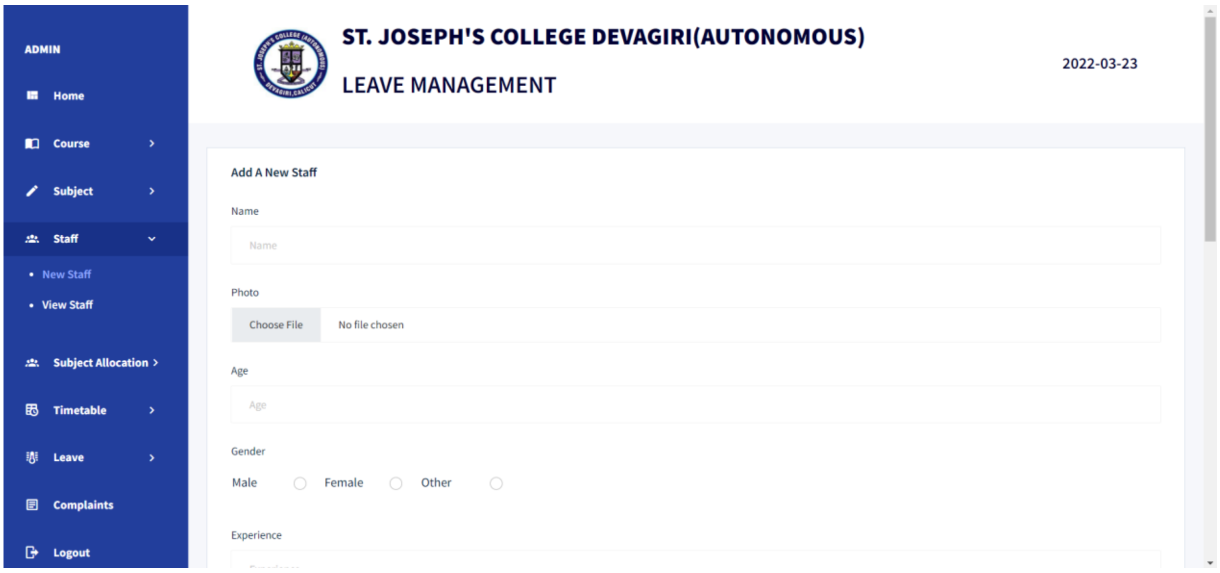
Course: Bsc.Computer Science

Subject: Subject

Subject Code: Course Code

Semester: 1

ADMIN ADD NEW STAFF



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

2022-03-23

Add A New Staff

Name: Name

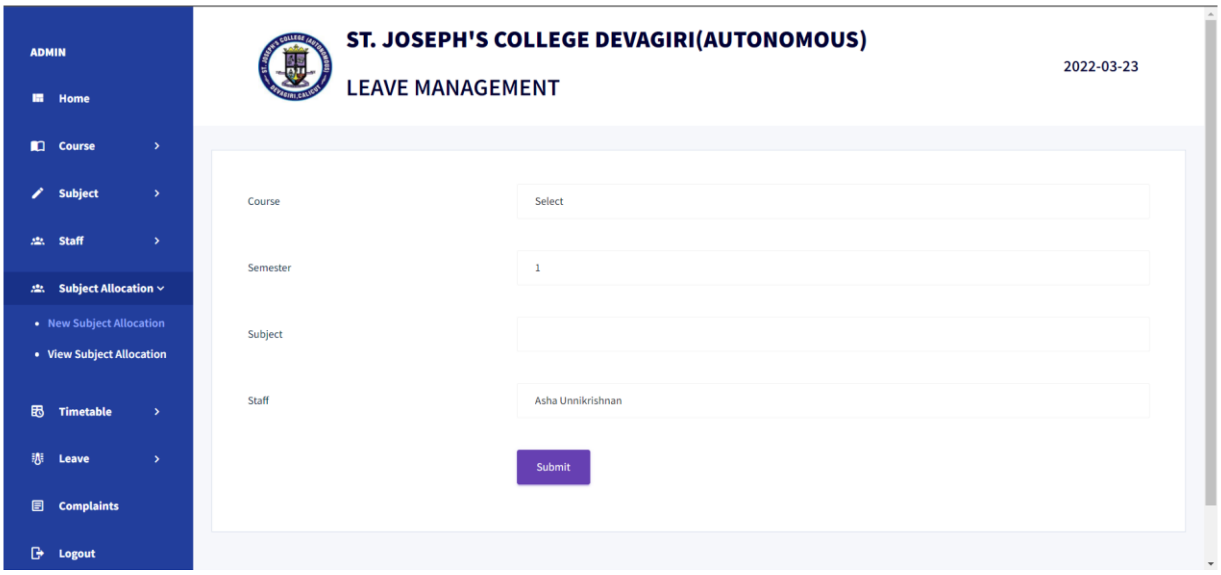
Photo: Choose File No file chosen

Age: Age

Gender: Male ○ Female ○ Other ○

Experience

ADMIN ADD NEW SUBJECT ALLOCATION



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

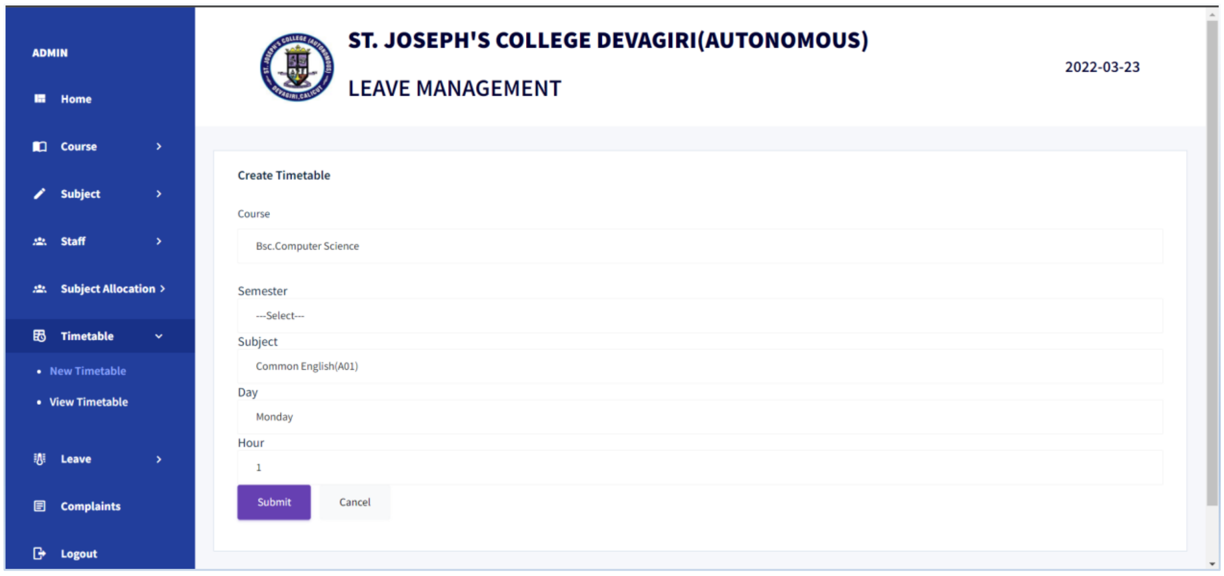
LEAVE MANAGEMENT

2022-03-23

Course	Select
Semester	1
Subject	
Staff	Asha Unnikrishnan

Submit

ADMIN ADD NEW TIMETABLE



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

2022-03-23

Create Timetable

Course: Bsc.Computer Science

Semester: ---Select---

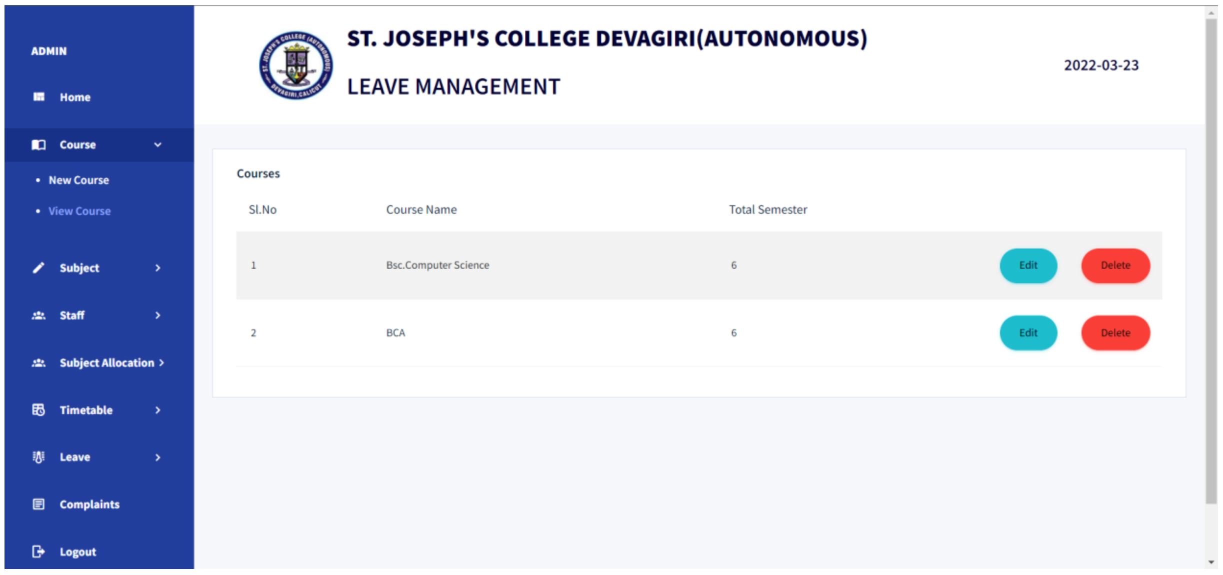
Subject: Common English(A01)

Day: Monday

Hour: 1

Submit Cancel

ADMIN VIEW COURSES



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

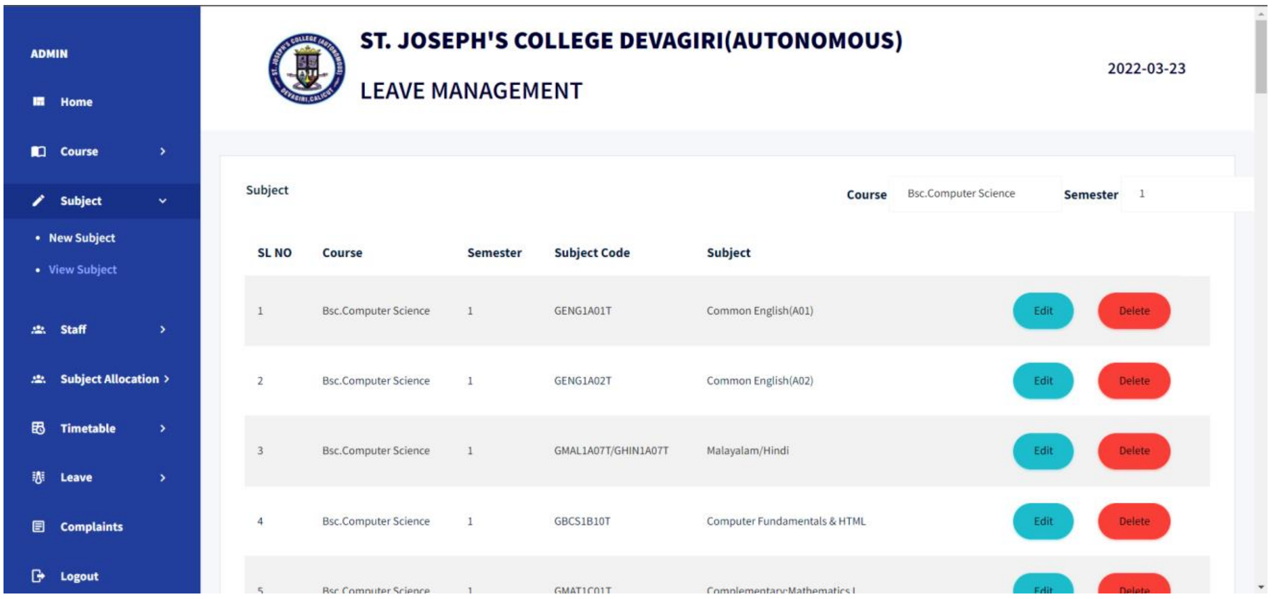
LEAVE MANAGEMENT

2022-03-23

Courses

Sl.No	Course Name	Total Semester		
1	Bsc.Computer Science	6	Edit	Delete
2	BCA	6	Edit	Delete

ADMIN VIEW SUBJECTS



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

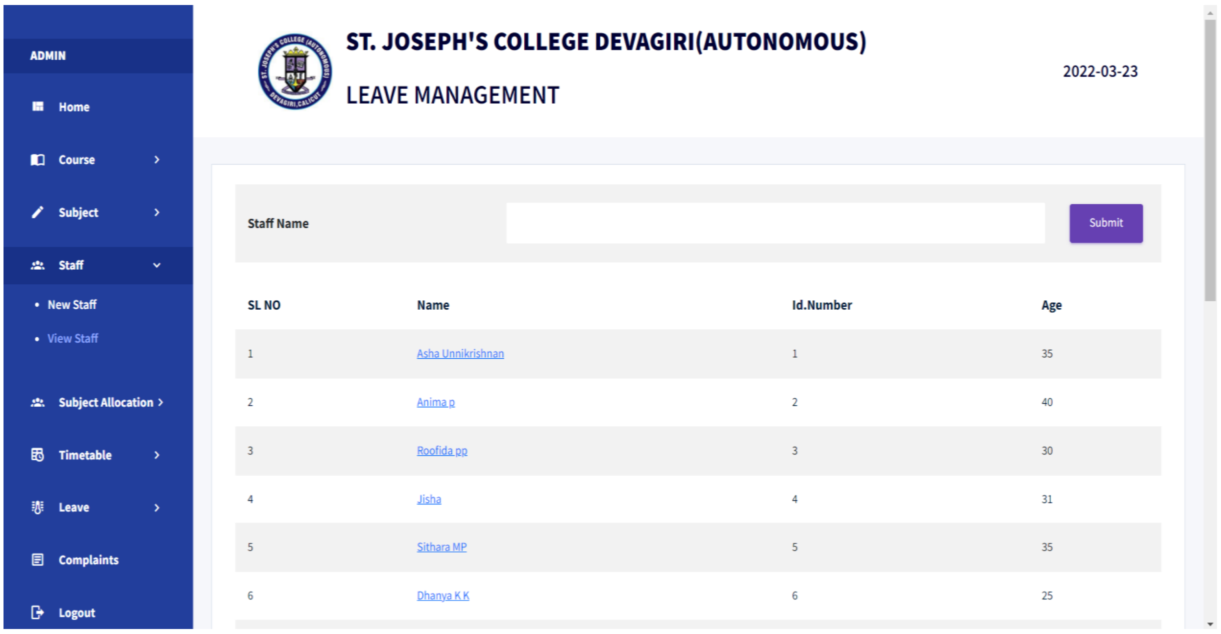
2022-03-23

Subject

Course: Bsc.Computer Science Semester: 1

SL NO	Course	Semester	Subject Code	Subject		
1	Bsc.Computer Science	1	GENG1A01T	Common English(A01)	Edit	Delete
2	Bsc.Computer Science	1	GENG1A02T	Common English(A02)	Edit	Delete
3	Bsc.Computer Science	1	GMAL1A07T/GHIN1A07T	Malayalam/Hindi	Edit	Delete
4	Bsc.Computer Science	1	GBCS1B10T	Computer Fundamentals & HTML	Edit	Delete
5	Bsc.Computer Science	1	GMAT1C01T	Complementary:Mathematics I	Edit	Delete

ADMIN VIEW STAFF



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

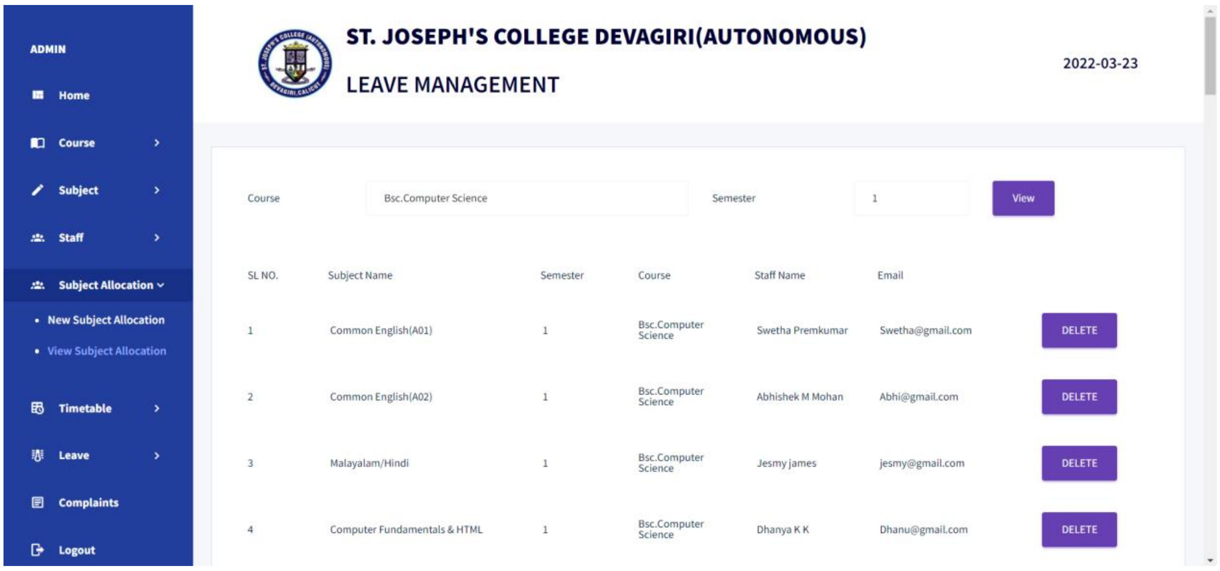
LEAVE MANAGEMENT

2022-03-23

Staff Name [ ] Submit

SL NO	Name	Id.Number	Age
1	Asha Unnikrishnan	1	35
2	Anima p	2	40
3	Roofida pp	3	30
4	Jisha	4	31
5	Sithara MP	5	35
6	Dhanya K K	6	25

ADMIN VIEW SUBJECT ALLOCATION



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

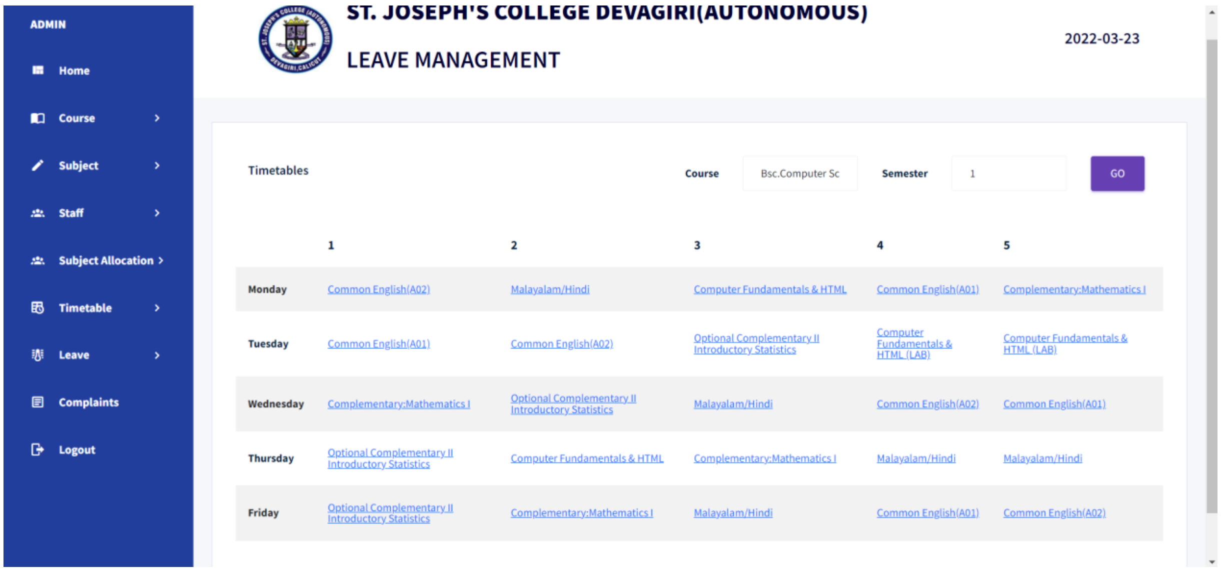
LEAVE MANAGEMENT

2022-03-23

Course: Bsc.Computer Science   Semester: 1   View

SL NO.	Subject Name	Semester	Course	Staff Name	Email	
1	Common English(A01)	1	Bsc.Computer Science	Swetha Premkumar	Swetha@gmail.com	DELETE
2	Common English(A02)	1	Bsc.Computer Science	Abhishek M Mohan	Abhi@gmail.com	DELETE
3	Malayalam/Hindi	1	Bsc.Computer Science	Jesmy james	jesmy@gmail.com	DELETE
4	Computer Fundamentals & HTML	1	Bsc.Computer Science	Dhanya K K	Dhanu@gmail.com	DELETE

ADMIN VIEW TIMETABLE



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

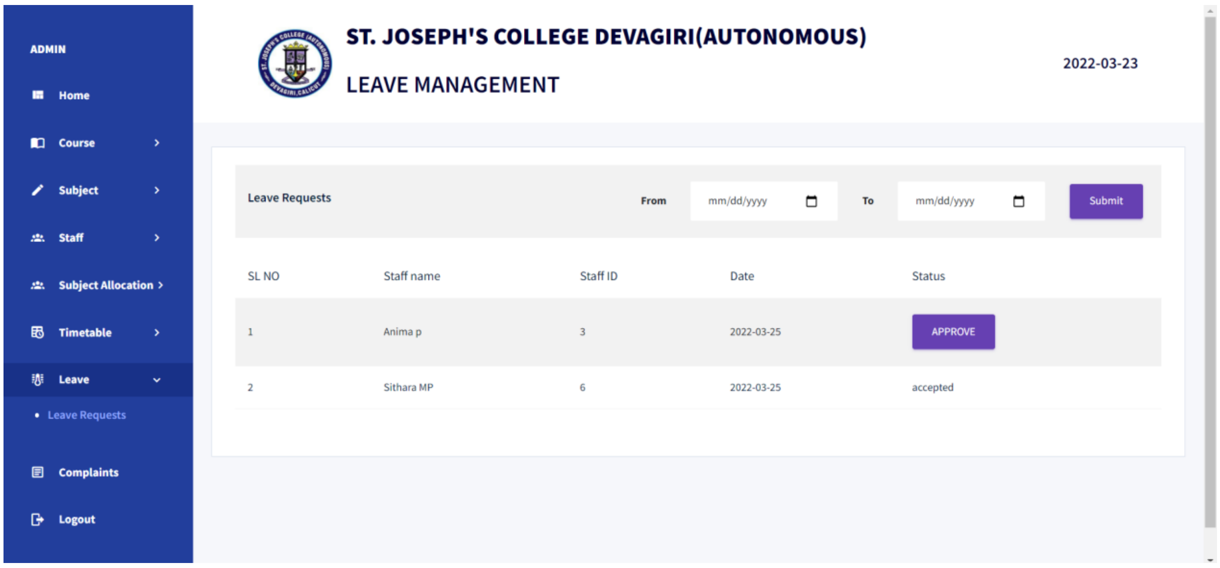
2022-03-23

Timetables

Course: Bsc.Computer Sc   Semester: 1   GO

	1	2	3	4	5
Monday	Common English(A02)	Malayalam/Hindi	Computer Fundamentals & HTML	Common English(A01)	Complementary:Mathematics I
Tuesday	Common English(A01)	Common English(A02)	Optional Complementary II Introductory Statistics	Computer Fundamentals & HTML (LAB)	Computer Fundamentals & HTML (LAB)
Wednesday	Complementary:Mathematics I	Optional Complementary II Introductory Statistics	Malayalam/Hindi	Common English(A02)	Common English(A01)
Thursday	Optional Complementary II Introductory Statistics	Computer Fundamentals & HTML	Complementary:Mathematics I	Malayalam/Hindi	Malayalam/Hindi
Friday	Optional Complementary II Introductory Statistics	Complementary:Mathematics I	Malayalam/Hindi	Common English(A01)	Common English(A02)

ADMIN VIEW LEAVE REQUESTS



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

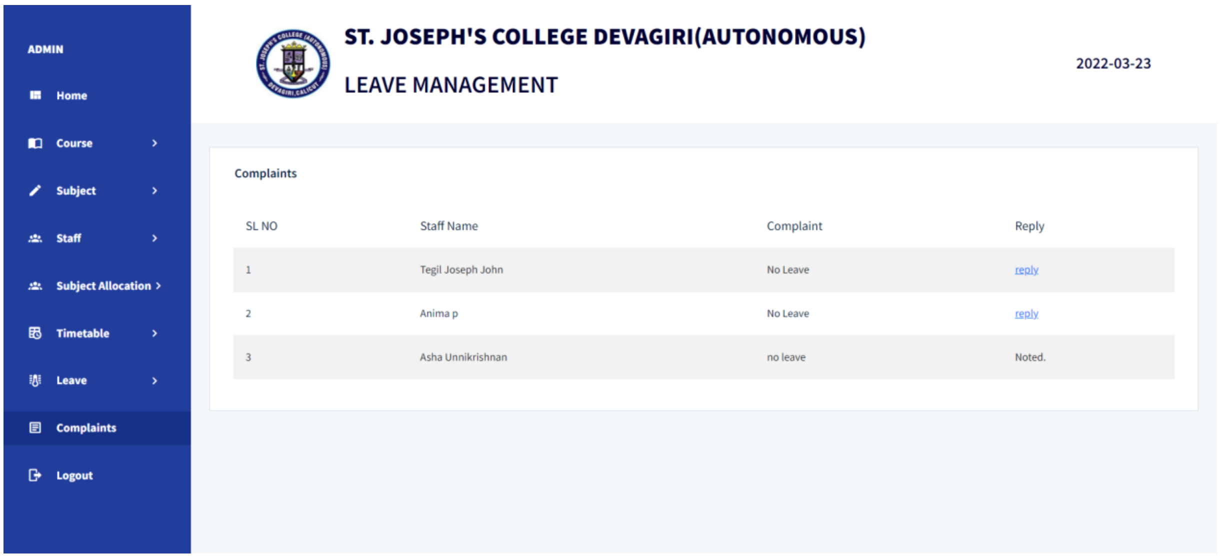
LEAVE MANAGEMENT

2022-03-23

Leave Requests From mm/dd/yyyy To mm/dd/yyyy Submit

SL NO	Staff name	Staff ID	Date	Status
1	Anima p	3	2022-03-25	APPROVE
2	Sithara MP	6	2022-03-25	accepted

ADMIN VIEW COMPLAINTS



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

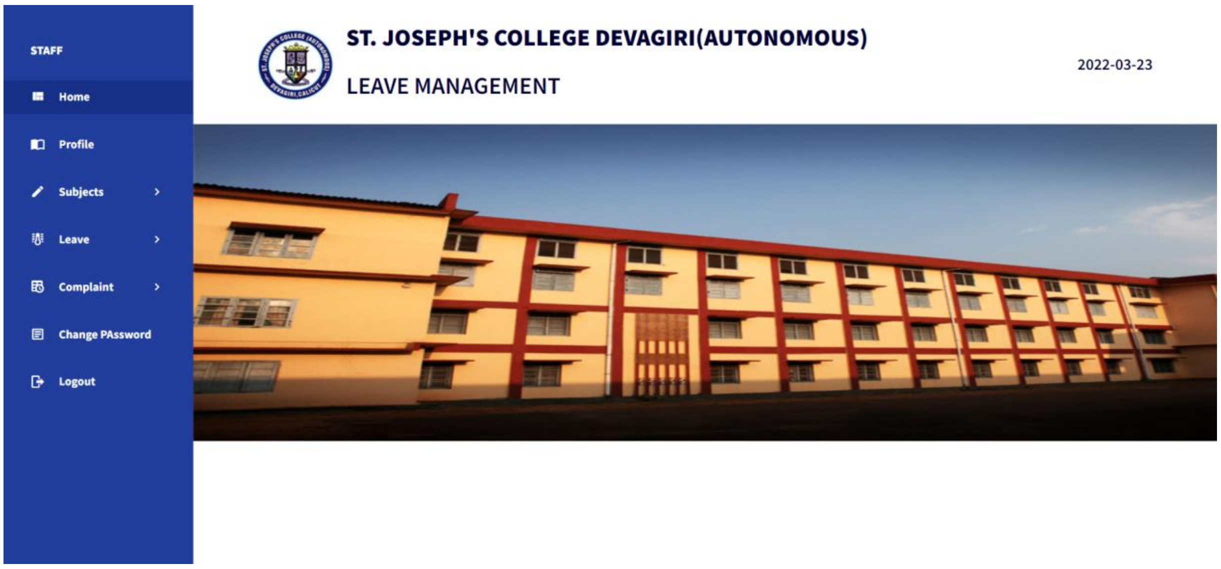
LEAVE MANAGEMENT

2022-03-23

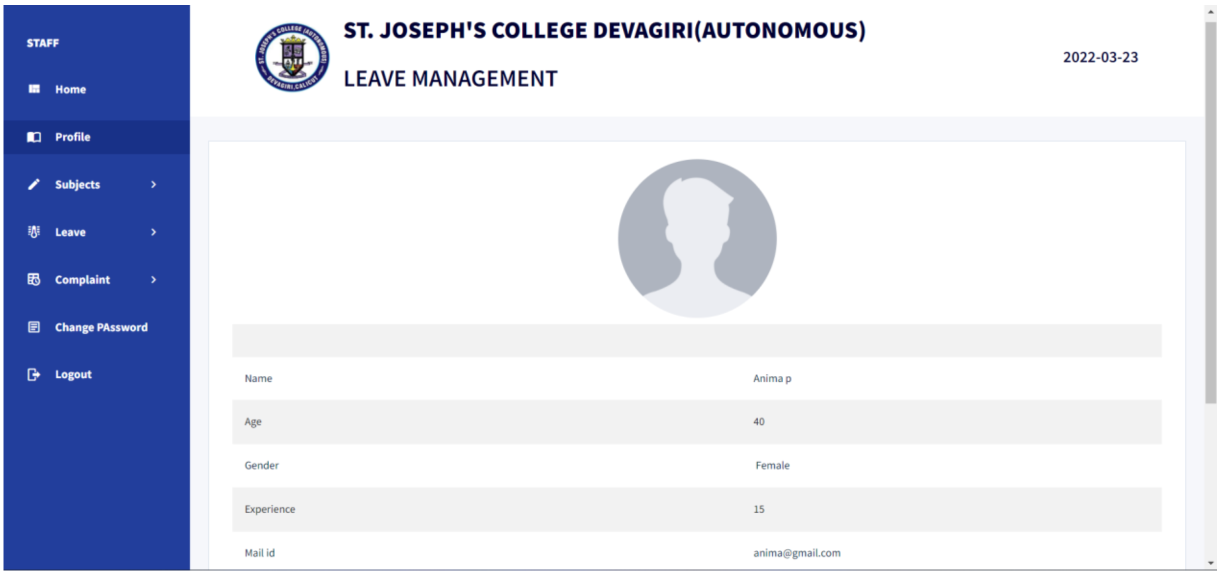
Complaints

SL NO	Staff Name	Complaint	Reply
1	Tegil Joseph John	No Leave	reply
2	Anima p	No Leave	reply
3	Asha Unnikrishnan	no leave	Noted.

STAFF HOMEPAGE

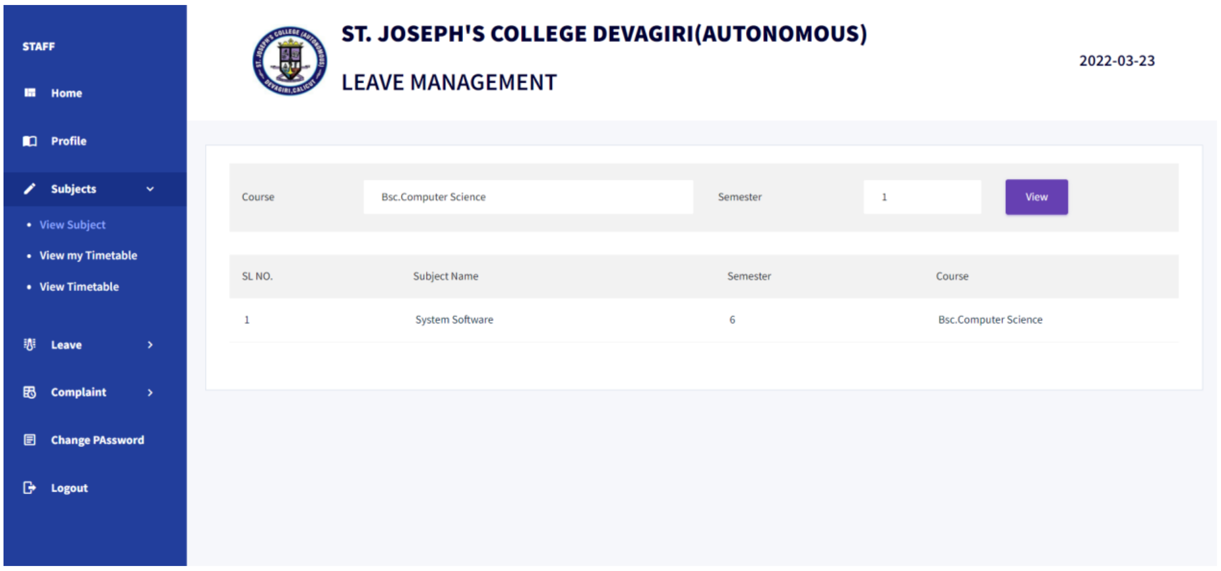


STAFF VIEW PROFILE



Name	Anima p
Age	40
Gender	Female
Experience	15
Mail id	anima@gmail.com

STAFF VIEW ALLOCATED SUBJECTS



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

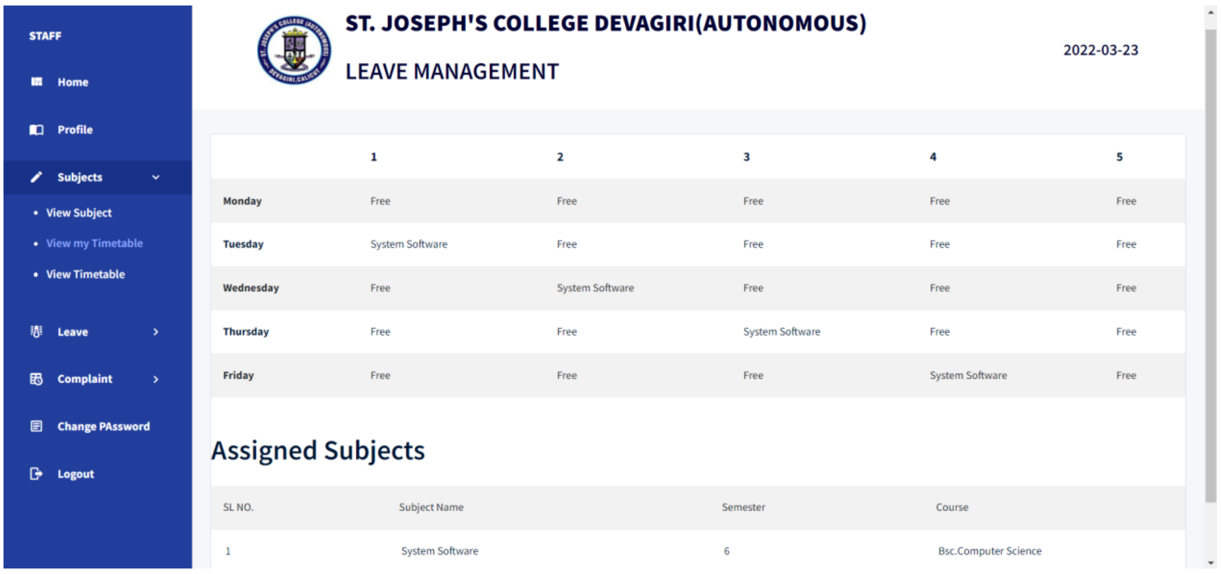
LEAVE MANAGEMENT

2022-03-23

Course: Bsc.Computer Science Semester: 1 View

SL NO.	Subject Name	Semester	Course
1	System Software	6	Bsc.Computer Science

STAFF VIEW PERSONAL TIMETABLE



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

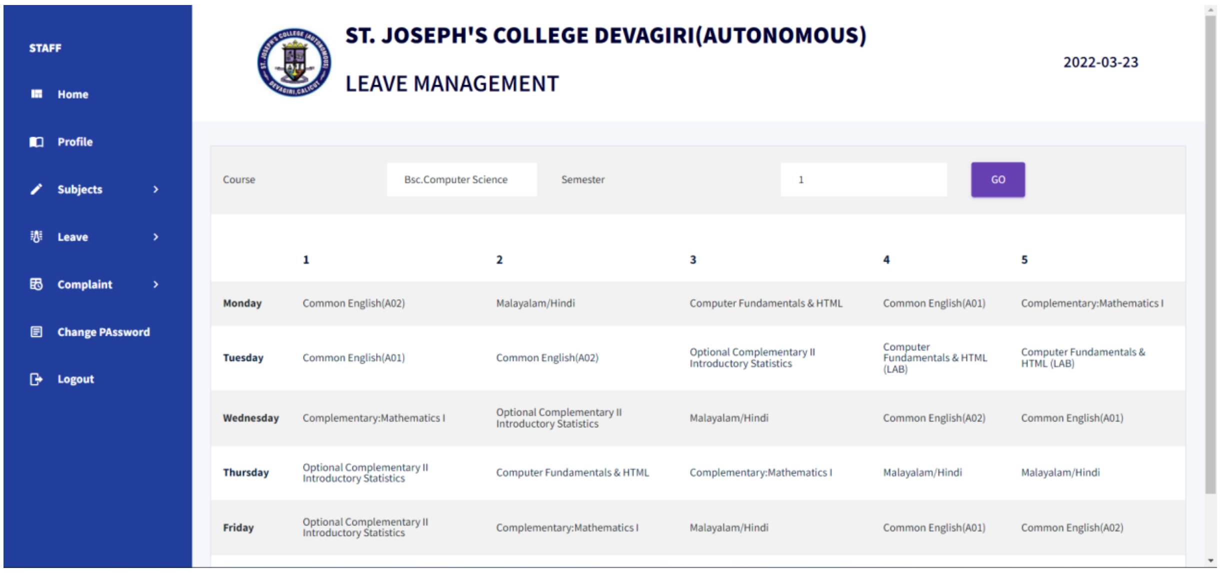
2022-03-23

	1	2	3	4	5
Monday	Free	Free	Free	Free	Free
Tuesday	System Software	Free	Free	Free	Free
Wednesday	Free	System Software	Free	Free	Free
Thursday	Free	Free	System Software	Free	Free
Friday	Free	Free	Free	System Software	Free

Assigned Subjects

SL NO.	Subject Name	Semester	Course
1	System Software	6	Bsc.Computer Science

STAFF VIEW TIMETABLE



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

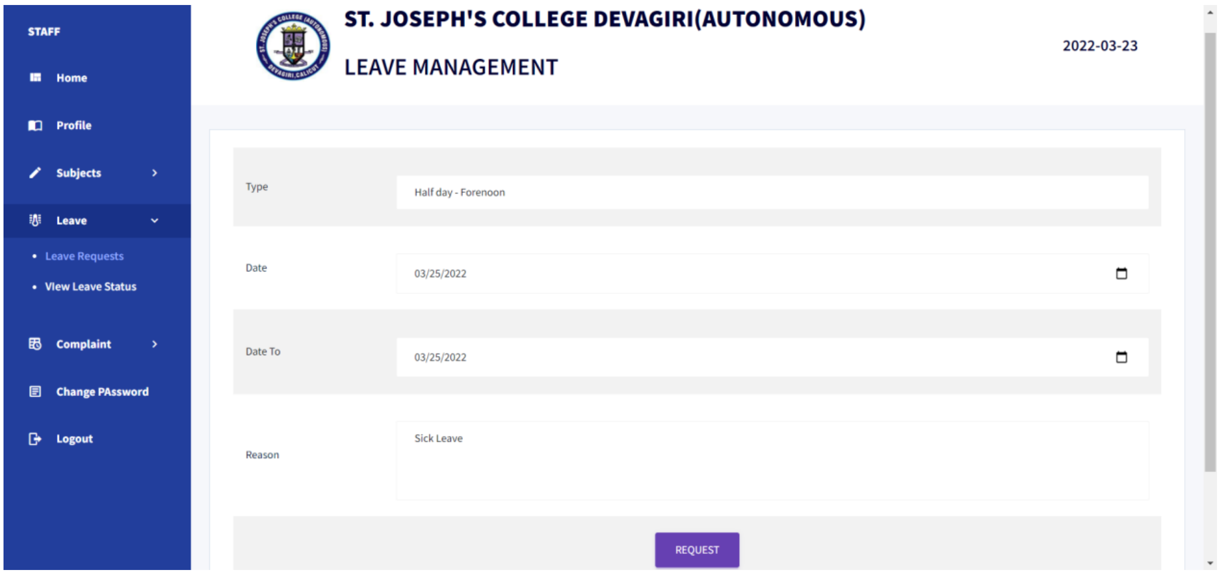
LEAVE MANAGEMENT

2022-03-23

Course Bsc.Computer Science Semester 1 GO

	1	2	3	4	5
Monday	Common English(A02)	Malayalam/Hindi	Computer Fundamentals & HTML	Common English(A01)	Complementary:Mathematics I
Tuesday	Common English(A01)	Common English(A02)	Optional Complementary II Introductory Statistics	Computer Fundamentals & HTML (LAB)	Computer Fundamentals & HTML (LAB)
Wednesday	Complementary:Mathematics I	Optional Complementary II Introductory Statistics	Malayalam/Hindi	Common English(A02)	Common English(A01)
Thursday	Optional Complementary II Introductory Statistics	Computer Fundamentals & HTML	Complementary:Mathematics I	Malayalam/Hindi	Malayalam/Hindi
Friday	Optional Complementary II Introductory Statistics	Complementary:Mathematics I	Malayalam/Hindi	Common English(A01)	Common English(A02)

STAFF REQUEST FOR LEAVE



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

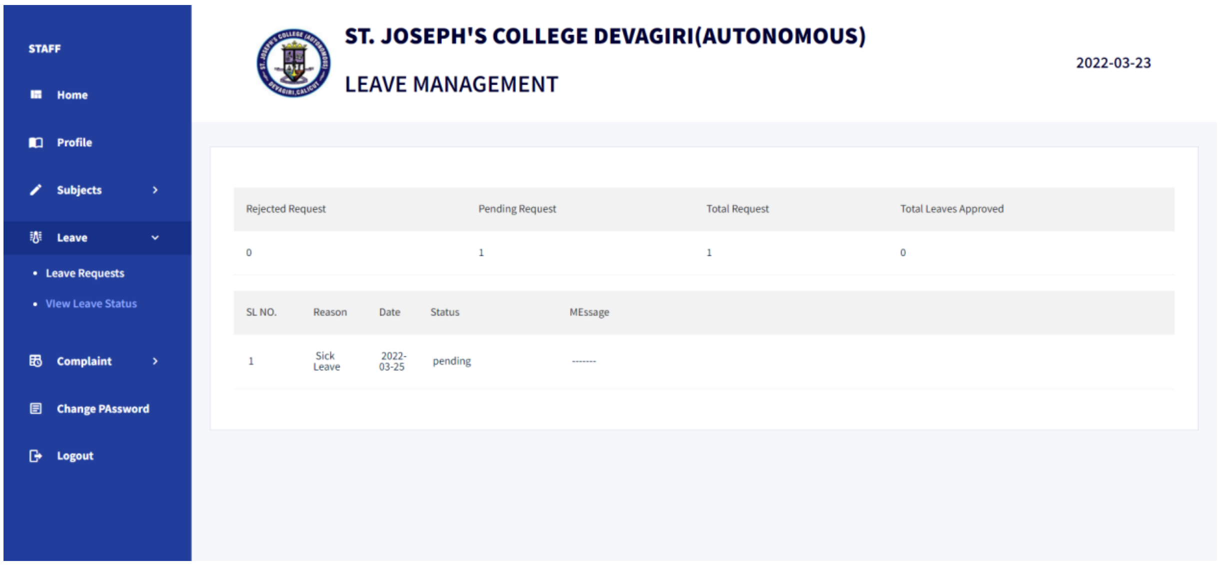
LEAVE MANAGEMENT

2022-03-23

Type	Half day - Forenoon
Date	03/25/2022
Date To	03/25/2022
Reason	Sick Leave

REQUEST

STAFF VIEW LEAVE STAFF



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

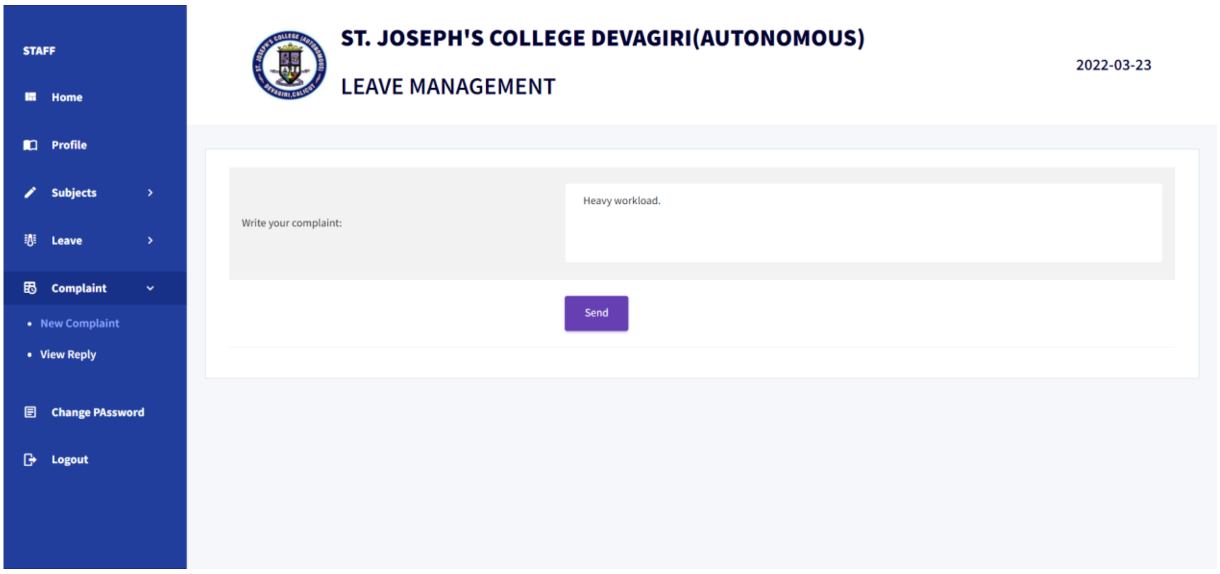
LEAVE MANAGEMENT

2022-03-23

Rejected Request	Pending Request	Total Request	Total Leaves Approved
0	1	1	0

SL NO.	Reason	Date	Status	MEssage
1	Sick Leave	2022-03-25	pending	-------

STAFF GIVE COMPLAINT



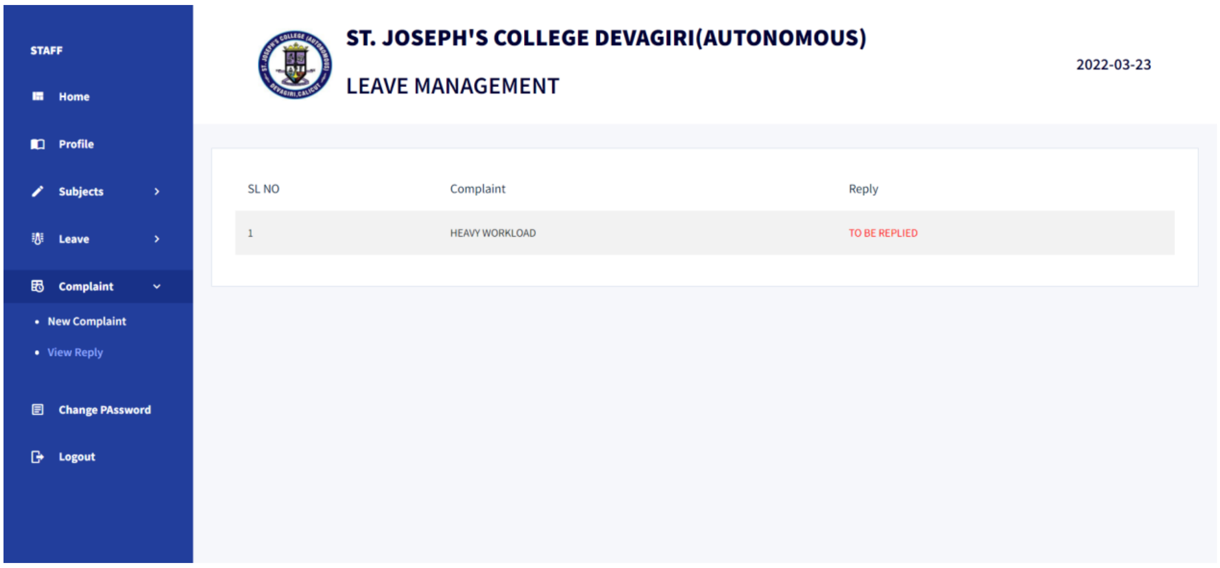
ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

2022-03-23

Write your complaint: Heavy workload.

STAFF VIEW REPLY



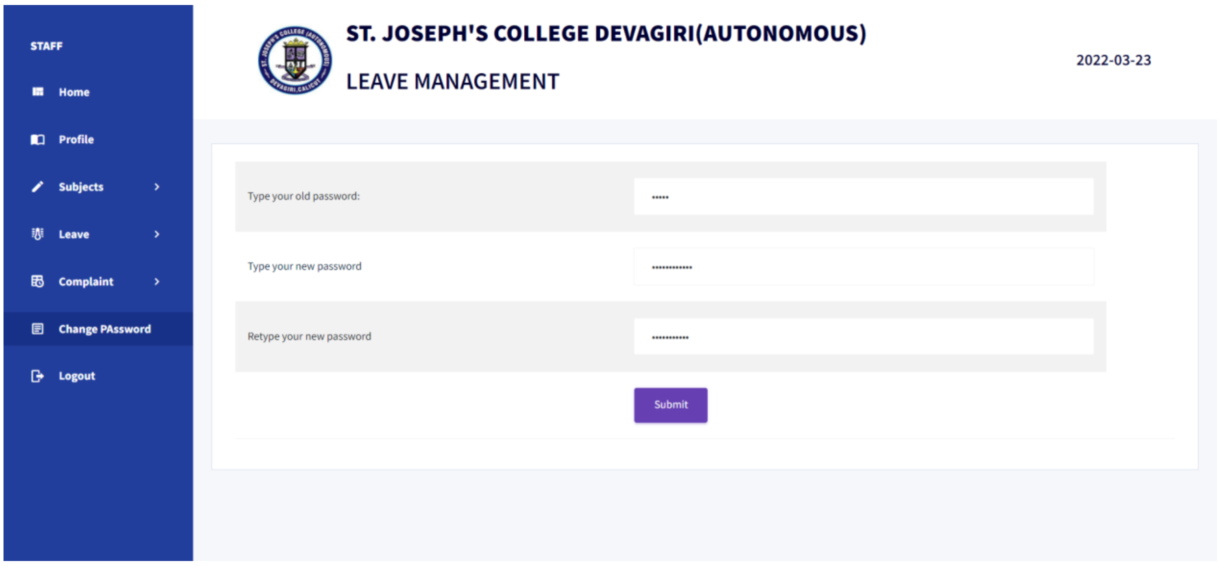
ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

2022-03-23

SL NO	Complaint	Reply
1	HEAVY WORKLOAD	TO BE REPLIED

STAFF CHANGE PASSWORD



ST. JOSEPH'S COLLEGE DEVAGIRI(AUTONOMOUS)

LEAVE MANAGEMENT

2022-03-23

Type your old password:	•••••
Type your new password	••••••••••••
Retype your new password	•••••••••••

2. SAMPLE SOURCE

```
from flask import Flask,render_template,request,session,jsonify

from flask_mail import Mail, Message
import time
import datetime
from DBConnection import Db
app = Flask(__name__)
app.secret_key="fff333"
mail= Mail(app)

app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'devagiricollege.leavemanagement@gmail.com'
app.config['MAIL_PASSWORD'] = 'devagirileavemanagement'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)




staticpath="C:\\Users\\Dell\\Desktop\\DVG\\project\\main\\untitled\\static\\"


@app.route('/')
def adminhome():
    return render_template('LOGIN PAGE.html')

@app.route('/login_post', methods=['post'])
def login_post():
    username=request.form['textfield']
    password=request.form['textfield2']
    db=Db()
    import datetime
    dd=str(datetime.datetime.today().now())
    ss=dd.split(" ")
    dd=ss[0]
    qry="select * from login where username='"+username+"' and
password='"+password+"'"
    res=db.selectOne(qry)
    session['l_id']=res['login_ID']
    session["dd"]=str(dd)
    if res is not None:
        type=res['type']
        if type=="admin":

            return render_template('admin/home.html')
        elif type=="teacher":
            return render_template('teacher/home.html')
        else:
            return 'invalid username or password'
    else:
        return 'invalid username or password'




@app.route('/adm_home')
def adm_home():
    return render_template('admin/home.html')
```

```
@app.route('/adm_add_course')
def adm_add_course():
    return render_template('admin/ADD COURSE.html')


@app.route('/adm_add_course_post',methods=['post'])
def adm_add_course_post():
    coursename=request.form['textfield']
    noofsem=request.form['textfield2']
    db=Db()
    qry="insert into
course(coursename,duration)values('"+coursename+"','"+noofsem+"')"
    print(qry)
    res=db.insert(qry)
    print(res)
    return render_template('admin/ADD COURSE.html')






@app.route('/admin_addsub')
def admin_addsub():
    c = Db()
    qry = "select * from course"
    res = c.select(qry)
    return render_template('admin/ADD SUB.html',data=res)

@app.route('/admin_addsub_post',methods=['post'])
def admin_addsub_post():
    coursename= request.form['select']
    subname=request.form['textfield2']
    subcode=request.form['textfield3']
    semester=request.form['select2']

    db = Db()
    qry = "insert into subject(course_name,subject_name,subcode,semester)values('" +
coursename+ "','"+subname+"','" + subcode + "','"+semester+"')"
    res = db.insert(qry)

    return admin_addsub()






@app.route('/teacherhome')
def teacherhome():
    return render_template('admin/teacher_profile.html')


@app.route('/admin_add_staff')
def admin_addstaff():
    return render_template('admin/add.staff.html')

@app.route('/admin_add_staff_post',methods=['post'])
def admin_addstaff_post():
    name=request.form['textfield']
    photo=request.files['fileField']
    age=request.form['textfield2']
    gender=request.form['RadioGroup1']
    Experience=request.form['textfield3']
    email=request.form['textfield4']
    phone=request.form['textfield5']
    house_name=request.form['textfield7']
    house_number=request.form['textfield7']
    city=request.form['textfield77']
    district=request.form['select']
    state=request.form['textfield10']
    pincode=request.form['textfield9']


    dt = time.strftime("%Y%m%d-%H%M%S")
    photo.save(staticpath+"staff_image\\"+dt+".jpg")
    path = "/static/staff_image/"+dt+".jpg"

    db = Db()
    qry2="insert into
login(username,password,type)values('"+email+"','"+phone+"','teacher')"
    res2=db.insert(qry2)

    qry = "insert into
staff(staff_name,experience,staff_photo,mail_id,phone_number,house_name,house_number
,city,district,pincode,state,login_ID,age,gender)
values('"+name+"','"+Experience+"','"+path+"','"+email+"','"+phone+"','"+house_name+
"','"+house_number+"','"+city+"','"+district+"','"+pincode+"','"+state+"','"+str(res
2)+"','"+age+"','"+gender+"')"
    res = db.insert(qry)
    print(res)

    return render_template('admin/add.staff.html')


@app.route('/admin_ADMIN_EDIT_COURSES/<id>')
def admin_admin_edit_courses(id):
    db=Db()
    qry="select * from course WHERE course_id = '"+str(id)+"'"
    res=db.selectOne(qry)
    print(res)
    return render_template('admin/ADMIN EDIT COURSE.html',data=res)

@app.route('/admin_ADMIN_DELETE_COURSES/<id>')
def admin_admin_delete_courses(id):
    db=Db()
    qry="delete from course WHERE course_id = '"+str(id)+"'"
    res=db.delete(qry)

    return "<script>alert('Deleted Successfully
');window.location='/admin_ADMIN_VIEW_COURSES'</script>"





@app.route('/admin_ADMIN_EDIT_COURSES_post',methods=['post'])
def admin_admin_edit_courses_post():
    id=request.form['iid']
    coursename=request.form['textfield']
    totsem=request.form['textfield2']
    db=Db()
    qry= "update course set coursename= '"+coursename+"',duration='"+totsem+"' WHERE
course_id ='"+id+"' "
    res=db.update(qry)
    return "<script>alert('Successfully
updated');window.location='/admin_ADMIN_VIEW_COURSES'</script>"

@app.route('/admin_edit_timetable/<id>')
def admin_admin_edit_time_table(id):
    c = Db()
    qry = "select * from course"
    res = c.select(qry)
    qry2 = "select * from subject"
    res2 = c.select(qry2)
    print(res2)
    qry5 = "select * from subject inner join time_table on
time_table.subject=subject.subject_id where tid='"+id+"'"
    d5 = c.selectOne(qry5)
    print(d5)
    return render_template('admin/Admin Edit Time
table.html',data=res,data2=res2,d=d5)

@app.route('/admin_admin_edit_time_table_post',methods=['post'])
def admin_admin_edit_time_table_post():
    course = request.form['select']
    semester = request.form['select2']
    subject = request.form['select3']
    day = request.form['select4']
    hour = request.form['select5']
    tid=request.form["tid"]

    db = Db()
    qry = "UPDATE `time_table` SET
`day`='"+day+"',`hour`='"+hour+"',`course`='"+course+"',`semester`='"+semester+"',`s
ubject`='"+subject+"' WHERE `tid`='"+tid+"'"
    print(qry)
    res = db.insert(qry)
    return admin_timetable()

    return "<script>alert('Successfully
updated');window.location='/admin_admin_view_time_table'</script>"



@app.route('/admin_ADMIN_VIEW_COURSES')
def admin_admin_view_courses():
    c=Db()
    qry="select * from course"
    res=c.select(qry)
    return render_template('admin/ADMIN VEIW COURSE.html',data=res)


@app.route('/admin_Admin_view_staff.html')
def admin_admin_view_staff():
    c = Db()
    qry = "select * from staff"
    res = c.select(qry)
    return render_template('admin/Admin view staff.html',data=res)


@app.route('/searchstaff',methods=['post'])
def searchstaff():
    search = request.form['select']
    c=Db()
    qry="select * from staff where staff_name like '%"+search+"%'"
    res = c.select(qry)
    return render_template('admin/Admin view staff.html', data=res)


@app.route('/admin_ADMIN_VIEW_COURSES_post',methods=['post'])
def admin_admin_view_courses_post():
    course=request.form['select']

    return render_template('admin/ADMIN VEIW COURSE.html')
```

```
@app.route('/admin_Admin_view_staff_post.html',methods=['post'])
def admin_admin_view_staff_post():
    name=request.form['select']
    return render_template('admin/Admin view staff.html')


@app.route('/admin_ADMIN_VIEW_STAFF2/<sid>')
def admin_admin_view_STAFF2(sid):
    qry="select *from staff where staff_id='"+sid+"'"
    c=Db()
    res=c.selectOne(qry)

    return render_template('admin/ADMIN VIEW STAFF2.html',res=res)

@app.route('/admin_ADMIN_EDIT_STAFF2/<sid>')
def admin_admin_edit_STAFF2(sid):
    qry="select *from staff where staff_id='"+sid+"'"
    c=Db()
    res=c.selectOne(qry)
    session["sid"]=sid
    return render_template('admin/admin_edit_staff.html',res=res)

@app.route('/admin_ADMIN_DELETE_STAFF2/<sid>')
def admin_admin_delete_staff2(sid):
    db=Db()
    qry="delete from staff WHERE staff_id = '"+str(sid)+"'"
    res=db.delete(qry)

    return "<script>alert('Deleted Successfully
');window.location='/admin_Admin_view_staff.html'</script>"

@app.route('/admin_ADMIN_EDIT_STAFF2_post',methods=['post'])
def admin_admin_edit_staff2_post():
    name = request.form['textfield']
    id=str(session["sid"])
    age = request.form['textfield2']
    gender = request.form['RadioGroup1']
    Experience = request.form['textfield3']
    email = request.form['textfield4']
    phone = request.form['textfield5']
    house_name = request.form['textfield7']
    house_number = request.form['textfield7']

    # -------------------
    city = request.form['textfield8']
    district = request.form['select']
    state = request.form['textfield10']
    pincode = request.form['textfield9']
    db=Db()

    if 'fileField' in request.files:
        photo = request.files['fileField']
        if photo.filename !="":

photo.save("C:\\Users\\Pranoy\\PycharmProjects\\untitled\\static\\staff_image\\"+pho
to.filename)
            path="/static/staff_image/"+photo.filename
            qry="update staff set
staff_name='"+name+"',experience='"+Experience+"',staff_photo='"+path+"',age='"+age+
"',gender='"+gender+"',mail_id='"+email+"',phone_number='"+phone+"',house_name='"+ho
use_name+"',house_number='"+house_number+"',city='"+city+"',district='"+district+"',
pincode='"+pincode+"',state='"+state+"' where staff_id='"+id+"'"
        else:
            qry = "update staff set staff_name='" + name + "',experience='" +
Experience + "',age='" + age + "',gender='" + gender + "',mail_id='" + email +
"',phone_number='" + phone + "',house_name='" + house_name + "',house_number='" +
house_number + "',city='" + city + "',district='" + district + "',pincode='" +
pincode + "',state='" + state + "' where staff_id='" + id + "'"

    else:
        qry = "update staff set staff_name='" + name + "',experience='" + Experience
+ "',age='" + age + "',gender='" + gender + "',mail_id='" + email +
"',phone_number='" + phone + "',house_name='" + house_name + "',house_number='" +
house_number + "',city='" + city + "',district='" + district + "',pincode='" +
pincode + "',state='" + state + "' where staff_id='" + id + "'"

    res=db.update(qry)
    return "<script>alert('Successfully
updated');window.location='/admin_Admin_view_staff.html'</script>"

@app.route("/get_subject_according_crs_and_sem")
def changev():
    d=Db()
    id=request.args.get('cid')
    sem = request.args.get('sem')
    query12 = "select * from subject where course_name='"+id+"' and
semester='"+sem+"'"
    print(query12)
    res=d.select(query12)
    print(res)
    return jsonify(res)
@app.route('/admin_admin_view_time_table')
def admin_admin_view_time_table():
    c = Db()
    qry = "select * from course"
    res = c.select(qry)


    return render_template('admin/Admin view Time table.html',
data=res,status="null")

@app.route('/admin_admin_view_time_table_post',methods=['post'])
def admin_admin_view_time_table_post():
    coursename=request.form['select']
    semester=request.form['select2']
    c = Db()
    qry = "select * from course"
    res = c.select(qry)
    qry1="select subject.subject_id,subject.subject_name,time_table.tid from subject
inner join time_table on time_table.subject=subject.subject_id where day='monday'
and time_table.course='"+coursename+"' and time_table.semester='"+semester+"'"
    d1=c.select(qry1)
    qry2 = "select subject.subject_id,subject.subject_name,time_table.tid from
subject inner join time_table on time_table.subject=subject.subject_id where
day='tuesday' and time_table.course='" + coursename + "' and time_table.semester='"
+ semester + "'"
    d2 = c.select(qry2)
    qry3 = "select subject.subject_id,subject.subject_name,time_table.tid from
subject inner join time_table on time_table.subject=subject.subject_id where
day='wednesday' and time_table.course='" + coursename + "' and
time_table.semester='" + semester + "'"
    d3 = c.select(qry3)
    print(d3)
    qry4 = "select subject.subject_id,subject.subject_name,time_table.tid from
subject inner join time_table on time_table.subject=subject.subject_id where
day='thursday' and time_table.course='" + coursename + "' and time_table.semester='"
+ semester + "'"
    d4 = c.select(qry4)
    qry5 = "select subject.subject_id,subject.subject_name,time_table.tid from
subject inner join time_table on time_table.subject=subject.subject_id where
day='friday' and time_table.course='" + coursename + "' and time_table.semester='" +
semester + "'"
    d5 = c.select(qry5)
    return render_template('admin/Admin view Time
table.html',data=res,d1=d1,d2=d2,d3=d3,d4=d4,d5=d5,status="full")




@app.route('/admin_admin_edit_staff')
def admin_admin_edit_staff():
    return render_template('admin/admin_edit_staff.html')

@app.route('/admin_admin_edit_staff_post',methods=['post'])
def admin_admin_edit_staff_post():
    name = request.form['textfield']
    photo = request.files['fileField']
    age = request.form['textfield2']
    gender = request.form['RadioGroup1']
    Experience = request.form['textfield3']
    email = request.form['textfield4']
    phone = request.form['textfield5']
    house_name=request.form['textfield12']
    house_number = request.form['textfield7']
    city = request.form['textfield8']
    district = request.form['select']
    state = request.form['textfield10']
    pincode = request.form['textfield9']

    db = Db()
    qry = "insert into
staff(staff_name,experience,photo,age,gender,mail_id,phone_number,house_name,house_n
umber,city,district,state,pincode)values('" + name + "','" +Experience+ "','"
+photo+"','" + age + "','" + gender +
"','"+email+"','"+phone+"','"+house_name+"','"+house_number+"','"+city+"','"+distric
t+"','"+state+"','"+pincode+"')"
    res = db.insert(qry)
    return render_template('admin/admin_edit_staff.html')

@app.route('/admin_admin_edit_sub')
def admin_admin_edit_sub():
    return render_template('admin/admin_edit_sub.html')

@app.route('/admin_admin_edit_sub_post',methods=['post'])
def admin_admin_edit_sub_post():
    course=request.form['select']
    semester=request.form['select2']
    subcode=request.form['textfield']
    subname=request.form['textfield2']
    return render_template('admin/admin_edit_sub.html')


@app.route('/admin_admin_requests')
def admin_admin_admin_requests():
    c=Db()
    qry="select staff.*,leave_req.* from leave_req,staff where
leave_req.staff_id=staff.login_id"
    res=c.select(qry)
    q="select * from staff"
    resa= c.select(qry)
    return render_template('admin/admin_requests.html',data=res,dataa= resa)




@app.route('/admin_admin_requests_post',methods=['post'])
def admin_admin_admin_requests_post():
    from_date=request.form['select']
    to=request.form['select2']
    # staffname=request.form['select3']
    c = Db()
    qry = "select staff.*,leave_req.* from leave_req,staff where
leave_req.staff_id=staff.login_id and date between '"+from_date+"' and '"+to+"' "
    res = c.select(qry)
    q = "select * from staff"
    resa = c.select(qry)
    return render_template('admin/admin_requests.html', data=res, dataa=resa)


@app.route('/admin_approve_requests_post/<id>/<sid>/<type>')
def admin_admin_approve_requests_post(id,sid,type):
    db=Db()

    qry = "update leave_req set status='accepted' where leave_request_id='" +
str(id) + "'"
    res = db.update(qry)
    qqq="SELECT * FROM `staff` WHERE `login_id`='"+sid+"'"
    dd=db.selectOne(qqq)
    emaaaail=dd["mail_id"]
    msg = Message('Hello', sender='devagiricollege.leavemanagement@gmail.com',
                  recipients=['nadhaashraf12@gmail.com',emaaaail])
    msg.body = "Hello " + dd["staff_name"] + ", Your leave has been approved. Please
Check it out !!!"
    mail.send(msg)
    if type =="f":
        qry1="SELECT date from leave_sub WHERE `leave_id`='"+id+"'"
        print(qry1)
        res1=db.select(qry1)
        for my in res1:
            dat=my["date"]

            qry2="SELECT DATE_FORMAT('"+str(dat)+"','%W')"
            dat2=db.selectOne(qry2)
            print(dat2)
            day=dat2["DATE_FORMAT('"+str(dat)+"','%W')"]
            print(day)
            day=str(day).lower()
            counter=0
            course_ids=[]
            sems=[]
            hrss=[]
            qry3="SELECT * FROM `subject` INNER JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER  JOIN `staff` ON
`staff`.`login_id`=`subject_staff_allocation`.`staff_id` INNER JOIN `time_table` ON
`time_table`.`subject`=`subject`.`subject_id` INNER JOIN course ON
`course`.`course_id`=`subject`.`course_name` WHERE
`subject_staff_allocation`.`staff_id`='"+sid+"' AND
`time_table`.`day`='"+str(day)+"'   "
            data3=db.select(qry3)
            # print(qry3)
            # print(data3)
            for i in data3:
                course_ids.append(str(i["course_id"]))
                sems.append(str(i["semester"]))
                hrss.append(str(i["hour"]))
            print(hrss)
            for i in range(len(sems)):
                qry4="SELECT * FROM SUBJECT INNER JOIN `subject_staff_allocation` on
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE
`subject`.`course_name`='"+course_ids[i]+"' AND `subject`.`semester`='"+sems[i]+"' "
                # print(qry4)
                data5 =db.select(qry4)
                for j in data5:
                    sub_id = str(j["subject_id"])
                    staff_id = str(j["staff_id"])
                    # print(data5)
                    qry7="SELECT COUNT(tid),`subject`.`subject_id` FROM `time_table`
INNER JOIN `subject` ON `subject`.`subject_id`=`time_table`.`subject` INNER JOIN
`subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE
`subject_staff_allocation`.`staff_id`='"+staff_id+"' and day='"+day+"'"
                    data6=db.selectOne(qry7)
                    # print(qry7)
                    # print(data6)
                    if data6 is not None:
                        count=data6["COUNT(tid)"]

                        print("aaaaaaaaa")
                        print(count)
                        if int(count)<4 and counter<len(sems):
                            qry8="SELECT * FROM `timetable_change` WHERE
`course_id`='"+course_ids[i]+"' AND `semester`='"+sems[i]+"' AND
`date`='"+str(day)+"' AND HOUR='"+hrss[i]+"'"
                            rr8=db.selectOne(qry8)

print(rr8,"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa")
                            print("bbb",counter,len(sems))
                            if rr8 is None:
                                qry6="INSERT INTO `timetable_change`
(`course_id`,`date`,`semester`,`hour`,`subject_id`)VALUES('"+course_ids[i]+"','"+str
(dat)+"','"+sems[i]+"','"+hrss[i]+"','"+sub_id+"') "
                                db.insert(qry6)
                                counter+=1

                                qqr="SELECT * FROM `timetable_change` INNER JOIN
`subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`timetable_change`.`subject_id` INNER JOIN
`staff` ON `staff`.`login_id`=`subject_staff_allocation`.`staff_id` where
login_id='"+staff_id+"'"
                                prog=db.selectOne(qqr)
                                if prog is not None:
                                    mail_id=prog["mail_id"]
                                    msg = Message('Hello',
sender='devagiricollege.leavemanagement@gmail.com',
recipients=['nadhaashraf12@gmail.com'])
                                    msg.body = "Hello "+ prog["staff_name"] +",
There is a change in "+ str(dat)+"'s Timetable. Please Check it out !!!"
                                    mail.send(msg)

                                break

                        # else:
                        #     break
                    # else:
                    #     continue

    else:
        qry1 = "SELECT date from leave_sub WHERE `leave_id`='" + id + "'"
        print(qry1)
        res1 = db.select(qry1)
        for my in res1:
            dat = my["date"]

            qry2 = "SELECT DATE_FORMAT('" + str(dat) + "','%W')"
            dat2 = db.selectOne(qry2)
            print(dat2)
            day = dat2["DATE_FORMAT('" + str(dat) + "','%W')"]
            print(day)
            day = str(day).lower()
            counter = 0
            course_ids = []
            sems = []
            hrss = []

            if type =="an":
                qry3 = "SELECT * FROM `subject` INNER JOIN
`subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER  JOIN `staff` ON
`staff`.`login_id`=`subject_staff_allocation`.`staff_id` INNER JOIN `time_table` ON
`time_table`.`subject`=`subject`.`subject_id` INNER JOIN course ON
`course`.`course_id`=`subject`.`course_name` WHERE
`subject_staff_allocation`.`staff_id`='" + sid + "' AND `time_table`.`day`='" +
str(day) + "' and time_table.hour>3  "
                data3 = db.select(qry3)
                print(qry3)
                # print(qry3)
                # print(data3)
                for i in data3:
                    course_ids.append(str(i["course_id"]))
                    sems.append(str(i["semester"]))
                    hrss.append(str(i["hour"]))
                print(hrss)
                for i in range(len(sems)):
                    qry4 = "SELECT * FROM SUBJECT INNER JOIN
`subject_staff_allocation` on
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE
`subject`.`course_name`='" + \
                           course_ids[i] + "' AND `subject`.`semester`='" + sems[i]
+ "' "
                    # print(qry4)
                    data5 = db.select(qry4)
                    for j in data5:
                        sub_id = str(j["subject_id"])
                        staff_id = str(j["staff_id"])
                        # print(data5)

                        qry7 = "SELECT COUNT(tid),`subject`.`subject_id` FROM
`time_table` INNER JOIN `subject` ON `subject`.`subject_id`=`time_table`.`subject`
INNER JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE
`subject_staff_allocation`.`staff_id`='" + staff_id + "' and day='" + day + "'"
                        data6 = db.selectOne(qry7)
                        # print(qry7)
                        # print(data6)
                        if data6 is not None:
                            count = data6["COUNT(tid)"]

                            print("aaaaaaaaa")
                            print(count)
                            if int(count) < 4 and counter < len(sems):
                                qry8 = "SELECT * FROM `timetable_change` WHERE
`course_id`='" + course_ids[
                                    i] + "' AND `semester`='" + sems[i] + "' AND
`date`='" + str(day) + "' AND HOUR='" + \
                                       hrss[i] + "'"
                                rr8 = db.selectOne(qry8)
                                print(rr8,
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa")
                                print("bbb", counter, len(sems))
                                if rr8 is None:
                                    qry6 = "INSERT INTO `timetable_change`
(`course_id`,`date`,`semester`,`hour`,`subject_id`)VALUES('" + \
                                           course_ids[i] + "','" + str(dat) + "','"
+ sems[i] + "','" + hrss[
                                               i] + "','" + sub_id + "') "
                                    db.insert(qry6)
                                    counter += 1

                                    qqr = "SELECT * FROM `timetable_change` INNER
```

```
JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`timetable_change`.`subject_id` INNER JOIN
`staff` ON `staff`.`login_id`=`subject_staff_allocation`.`staff_id` where
login_id='" + staff_id + "'"
                                    prog = db.selectOne(qqr)
                                    if prog is not None:
                                        mail_id = prog["mail_id"]
                                        msg = Message('Hello',
sender='devagiricollege.leavemanagement@gmail.com',

recipients=['nadhaashraf12@gmail.com'])
                                        msg.body = "Hello " + prog["staff_name"] +
", There is a change in " + str(
                                            dat) + "'s Timetable. Please Check it
out !!!"
                                        mail.send(msg)

                                    break

                                    # else:
                                    #     break
                                    # else:
                                    #     continue

            else:
                qry3 = "SELECT * FROM `subject` INNER JOIN
`subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER  JOIN `staff` ON
`staff`.`login_id`=`subject_staff_allocation`.`staff_id` INNER JOIN `time_table` ON
`time_table`.`subject`=`subject`.`subject_id` INNER JOIN course ON
`course`.`course_id`=`subject`.`course_name` WHERE
`subject_staff_allocation`.`staff_id`='" + sid + "' AND `time_table`.`day`='" + str(
                    day) + "' and time_table.hour<3  "
                data3 = db.select(qry3)
                # print(qry3)
                # print(data3)
                for i in data3:
                    course_ids.append(str(i["course_id"]))
                    sems.append(str(i["semester"]))
                    hrss.append(str(i["hour"]))
                print(hrss)
                for i in range(len(sems)):
                    qry4 = "SELECT * FROM SUBJECT INNER JOIN
`subject_staff_allocation` on
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE
`subject`.`course_name`='" + \
                           course_ids[i] + "' AND `subject`.`semester`='" + sems[i]
+ "' "
                    # print(qry4)
                    data5 = db.select(qry4)
                    for j in data5:
                        sub_id = str(j["subject_id"])
                        staff_id = str(j["staff_id"])
                        # print(data5)

                        qry7 = "SELECT COUNT(tid),`subject`.`subject_id` FROM
`time_table` INNER JOIN `subject` ON `subject`.`subject_id`=`time_table`.`subject`
INNER JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE
`subject_staff_allocation`.`staff_id`='" + staff_id + "' and day='" + day + "'"
                        data6 = db.selectOne(qry7)
                        # print(qry7)
                        # print(data6)
                        if data6 is not None:
                            count = data6["COUNT(tid)"]

                            print("aaaaaaaaa")
                            print(count)
                            if int(count) < 4 and counter < len(sems):
                                qry8 = "SELECT * FROM `timetable_change` WHERE
`course_id`='" + course_ids[
                                    i] + "' AND `semester`='" + sems[i] + "' AND
`date`='" + str(day) + "' AND HOUR='" + \
                                       hrss[i] + "'"
                                rr8 = db.selectOne(qry8)
                                print(rr8,
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa")
                                print("bbb", counter, len(sems))
                                if rr8 is None:
                                    qry6 = "INSERT INTO `timetable_change`
(`course_id`,`date`,`semester`,`hour`,`subject_id`)VALUES('" + \
                                           course_ids[i] + "','" + str(dat) + "','"
+ sems[i] + "','" + hrss[
                                               i] + "','" + sub_id + "') "
                                    db.insert(qry6)
                                    counter += 1

                                    qqr = "SELECT * FROM `timetable_change` INNER
JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`timetable_change`.`subject_id` INNER JOIN
`staff` ON `staff`.`login_id`=`subject_staff_allocation`.`staff_id` where
login_id='" + staff_id + "'"
                                    prog = db.selectOne(qqr)
                                    if prog is not None:
                                        mail_id = prog["mail_id"]
                                        msg = Message('Hello',
sender='devagiricollege.leavemanagement@gmail.com',

recipients=['nadhaashraf12@gmail.com'])
                                        msg.body = "Hello " + prog["staff_name"] +
", There is a change in " + str(
                                            dat) + "'s Timetable. Please Check it
out !!!"
                                        mail.send(msg)

                                    break

                                    # else:
                                    #     break
                                    # else:
                                    #     continue


    return '''<script>alert('accepted'); window.location='/admin_admin_requests'
</script>'''




@app.route('/admin_reject_requests_post/<id>')
def admin_admin_reject_requests_post(id):
    db=Db()

    return render_template("admin/Message.html",id=id)



@app.route('/admin_reject_requests_post_post',methods=['POST'])
def admin_reject_requests_post_post():
    db=Db()
    message=request.form["textfield"]
    id=request.form["ii"]
    qry="update leave_req set status='rejected',message='"+message+"' where
leave_request_id='"+str(id)+"'"
```

```
    res=db.update(qry)
    return '''<script>alert('rejected'); window.location='/admin_admin_requests'
</script>'''


@app.route('/admin_ADMIN_VIEW_ALLOCATE')
def admin_admin_view_allocate():
    db=Db()
    qry="select * from course"
    res=db.select(qry)

    qry1="SELECT * FROM `staff` INNER JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`staff_id`=`staff`.`login_id` INNER JOIN `subject` ON
`subject`.`subject_id`=`subject_staff_allocation`.`subject_id` INNER JOIN `course`
ON `course`.`course_id`=`subject`.`course_name`"
    res1=db.select(qry1)
    return render_template('admin/ADMIN_VIEW_ALLOCATE.html',data=res,data1=res1)

@app.route('/admin_ADMIN_VIEW_ALLOCATE_post',methods=['post'])
def admin_admin_view_allocate_post():
    course=request.form['select']
    sem=request.form['select2']
    db = Db()
    qry = "select * from course"
    res = db.select(qry)

    qry1 = "SELECT * FROM `staff` INNER JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`staff_id`=`staff`.`login_id` INNER JOIN `subject` ON
`subject`.`subject_id`=`subject_staff_allocation`.`subject_id` INNER JOIN `course`
ON `course`.`course_id`=`subject`.`course_name` where course_id='"+course+"' and
semester='"+sem+"'"
    res1 = db.select(qry1)
    return render_template('admin/ADMIN_VIEW_ALLOCATE.html', data=res, data1=res1)



@app.route("/admin_ADMIN_delete_ALLOCATE/<id>")
def aa(id):
    db = Db()
    qry = "delete from subject_staff_allocation where satff_subject_id='"+id+"'"
    res = db.delete(qry)
    return admin_admin_view_allocate()



@app.route('/admin_admin_view_sub')
def admin_admin_view_sub():
    c = Db()
    qry = "select subject.*,subject.semester as sem,course.* from subject,course
where course.course_id=subject.course_name"
    res2 = c.select(qry)
    qry = "select * from course"
    res = c.select(qry)
    return render_template('admin/Admin_view_sub.html', data=res2, data1=res)





@app.route('/admin_admin_view_sub_post',methods=['post'] )
def admin_admin_view_sub_post():
    course = request.form['select']
    sem = request.form['select2']
    c = Db()
    qry = "select subject.*,subject.semester as sem,course.* from subject inner join
course on course.course_id=subject.course_name where
subject.course_name='"+str(course)+"' and subject.semester='"+str(sem)+"'"
```

```
    res2 = c.select(qry)
    qry = "select * from course"
    res = c.select(qry)
    return render_template('admin/Admin_view_sub.html', data=res2, data1=res)



@app.route('/admin_admin_view_sub_del/<v>')
def admin_admin_view_sub_del(v):
    c = Db()
    qry = "delete from subject where subject_id='"+v+"'"
    res2 = c.delete(qry)
    return admin_admin_view_sub()

@app.route('/admin_admin_view_sub_edit/<v>')
def admin_admin_view_sub_edit(v):
    c = Db()
    qry="select * from subject where subject_id='"+v+"'"
    res=c.selectOne(qry)
    qry2="select course.* from course,subject where
subject.course_name=course.course_id and subject.subject_id='"+v+"'"
    res2=c.selectOne(qry2)
    qry3="select * from course"
    res3=c.select(qry3)
    return
render_template("admin/admin_edit_sub.html",data=res,data2=res2,data3=res3)




@app.route('/admin_editsub_post',methods=['post'])
def admin_editsub_post():
    coursename= request.form['select']
    subname=request.form['textfield2']
    subcode=request.form['textfield']
    semester=request.form['select2']
    subid=request.form['subid']
    db = Db()
    print(coursename,subname,subcode,semester)
    qry = "update subject set
course_name='"+coursename+"',subject_name='"+subname+"',subcode='"+subcode+"',semest
er='"+semester+"' where subject_id='"+subid+"'"
    res = db.update(qry)
    return admin_admin_view_sub()




@app.route('/admin_ALLOCATE')
def admin_Allocate():
    db=Db()
    qry="select * from course"
    res=db.select(qry)
    qry = "select * from staff"
    res1 = db.select(qry)
    return render_template('admin/ALLOCATE.html',data=res,dd=res1)

@app.route('/admin_ALLOCATE_post',methods=['post'])
def admin_Allocate_post():

    subject=request.form['select3']
    staff=request.form['select4']
    bb="SELECT * FROM `subject_staff_allocation` WHERE subject_id='"+subject+"'"
    db=Db()
    res=db.selectOne(bb)
    if res is not None:
        satff_subject_id=str(res['satff_subject_id'])
        up="UPDATE `subject_staff_allocation` SET `staff_id`='"+staff+"' WHERE
`satff_subject_id`='"+satff_subject_id+"'"
        db.update(up)
    else:
        qry="INSERT INTO `subject_staff_allocation`
(`subject_id`,`staff_id`)VALUES('"+subject+"','"+staff+"') "
        db.insert(qry)
    return admin_Allocate()

@app.route('/adminviewcomplaint')
def adminviewcomplaint():
    db=Db()
    qry="select complaint.*,staff.* from staff inner join complaint on
staff.staff_id=complaint.staff_id"
    res = db.select(qry)
    print(res)
    return render_template('admin/adminveiwcomplaint.html',data=res)

@app.route('/admin_Reply_complain/<id>')
def admin_Reply_complaint(id):
    db = Db()
    qry = "select complaint.*,staff.* from staff inner join complaint on
staff.staff_id=complaint.staff_id Where complaint.complaint_id='"+id+"'"
    res = db.selectOne(qry)
    return render_template('admin/Reply complaint.html', rep=res)

@app.route('/admin_Reply_complain_post',methods=['post'])
def admin_Reply_complaint_post():
    cid = request.form['c_id']
    reply=request.form['textarea']
    db = Db()
    qry = "update complaint set reply='"+reply+"', status='replied' where
complaint_id='"+cid+"'"
    res = db.update(qry)
    return
'''<script>alert('Send');window.location='/adminviewcomplaint'</script>'''
@app.route('/admin_timetable')
def admin_timetable():
    c=Db()
    qry="select * from course"
    res=c.select(qry)

    qry2 = "select * from subject"
    res2 = c.select(qry2)
    print(res2)
    return render_template('admin/timetable.html',data=res,data2=res2)

@app.route('/admin_timetable',methods=['post'])
def admin_timetable_post():
    course=request.form['select']
    semester=request.form['select2']
    subject=request.form['select3']
    day=request.form['select4']
    hour=request.form['select5']

    db = Db()
    qry ="insert into time_table(day,hour,course,semester,subject)VALUES
('"+day+"','"+hour+"','"+course+"','"+semester+"','"+subject+"')"
    print(qry)
    res = db.insert(qry)
    return admin_timetable()




#___________________________________________________________________________
```

```
@app.route('/teacher_change_password')
def teacher_change_password():
    return render_template('teacher/Teacher change password.html')

@app.route('/teacher_change_password_post',methods=['post'])
def teacher_change_password_post():
    newpassword=request.form['textfield']
    password2=request.form['textfield2']
    password1=request.form['textfield3']
    db=Db()
    qry="select * from login where password='"+password1+"' and
login_ID='"+str(session['l_id'])+"' "
    res=db.selectOne(qry)
    if res!=None:
        qry1="update login set password='"+password2+"' where
login_ID='"+str(session['l_id'])+"' "
        res1=db.update(qry1)
    return render_template('teacher/Teacher change password.html')

@app.route('/Teacher_request_leave')
def teacher_request_leave():
    return render_template('teacher/TEACHER REQUEST LEAVE.HTML')

@app.route('/Teacher_request_leave_post',methods=['post'])
def teacher_request_leave_post():
    db = Db()
    type=request.form["ss"]
    date_to=request.form['select2']
    reason = request.form['textarea']
    if type == "fn" :
        ff1=request.form["select25"]


        qry = "insert into leave_req(date,reason,status,staff_id,date_to,TYPE
)values('"+date_to+"','"+reason+"','pending','"+str(session["l_id"])+"','"+ff1+"','"
+type+"')"
        res = db.insert(qry)
        qqrr="SELECT DATEDIFF(date_to,date) as d FROM `leave_req` WHERE
`leave_request_id`='"+str(res)+"'"
        print(qqrr)
        ff=db.selectOne(qqrr)
        count=int(ff["d"])

        for i in range(count+1):
            if i == 0:
                rdate=date_to
                qrt="INSERT INTO `leave_sub`(`date`,`leave_id`)
VALUES('"+rdate+"','"+str(res)+"')"
                db.insert(qrt)
            else:
                q1 = "SELECT DATE_ADD('" +rdate+"', INTERVAL 1 DAY) as a;  "
                rr=db.selectOne(q1)
                rdate=rr['a']
                qrt = "INSERT INTO `leave_sub`(`date`,`leave_id`) VALUES('" + rdate
+ "','" + str(res) + "')"
                db.insert(qrt)
    else:
        qry = "insert into leave_req(date,reason,status,staff_id,date_to,TYPE
)values('" + date_to + "','" + reason + "','pending','" + str(session["l_id"]) +
"','" + date_to + "','" + type + "')"
        res = db.insert(qry)
        qrt = "INSERT INTO `leave_sub`(`date`,`leave_id`) VALUES('" + date_to +
"','" + str(res) + "')"
        db.insert(qrt)
    return
'''<script>alert('Success');window.location='/Teacher_request_leave'</script>'''

@app.route('/teacher_send_complaint')
def teacher_send_complaint():
    return render_template('teacher/Teacher send complaint.html')

@app.route('/teacher_send_complaint_post',methods=['post'])
def teacher_send_complaint_post():
    write_complaint=request.form['textarea']
    qry="INSERT INTO
`complaint`(`staff_id`,`date`,`complaint`,`reply`,`status`)VALUES('"+str(session["l_
id"])+"',CURDATE(),'"+write_complaint+"','pending','pending')"
    db=Db()
    db.insert(qry)
    return
'''<script>alert('Success');window.location='/teacher_send_complaint'</script>'''

@app.route('/teacher_view_timetable')
def teacher_view_timetable():
    c = Db()
    qry = "select * from course"
    res = c.select(qry)

    qrr="SELECT * FROM `subject` INNER JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER JOIN
`timetable_change` ON `timetable_change`.`subject_id`=`subject`.`subject_id` INNER
JOIN `course` ON `course`.`course_id`=`subject`.`course_name`WHERE
`timetable_change`.date>CURDATE() AND
`subject_staff_allocation`.`staff_id`='"+str(session["l_id"])+"'"
    db=Db()
    re=db.select(qrr)
    print(re)
    return render_template('teacher/Teacher veiw
Timeable.html',data=res,status="null",ll=len(re),re=re)

@app.route('/teacher_view_timetable_post',methods=['post'])
def teacher_view_timetable_post():
    coursename=request.form['select']
    semester=request.form['select2']
    c = Db()
    qry = "select * from course"
    res = c.select(qry)
    qry1="select subject.subject_id,subject.subject_name,time_table.tid from subject
inner join time_table on time_table.subject=subject.subject_id where day='monday'
and time_table.course='"+coursename+"' and time_table.semester='"+semester+"'"
    d1=c.select(qry1)
    qry2 = "select subject.subject_id,subject.subject_name,time_table.tid from
subject inner join time_table on time_table.subject=subject.subject_id where
day='tuesday' and time_table.course='" + coursename + "' and time_table.semester='"
+ semester + "'"
    d2 = c.select(qry2)
    qry3 = "select subject.subject_id,subject.subject_name,time_table.tid from
subject inner join time_table on time_table.subject=subject.subject_id where
day='wednesday' and time_table.course='" + coursename + "' and
time_table.semester='" + semester + "'"
    d3 = c.select(qry3)
    print(d3)
    qry4 = "select subject.subject_id,subject.subject_name,time_table.tid from
```

```
subject inner join time_table on time_table.subject=subject.subject_id where
day='thursday' and time_table.course='" + coursename + "' and time_table.semester='"
+ semester + "'"
    d4 = c.select(qry4)
    qry5 = "select subject.subject_id,subject.subject_name,time_table.tid from
subject inner join time_table on time_table.subject=subject.subject_id where
day='friday' and time_table.course='" + coursename + "' and time_table.semester='" +
semester + "'"
    d5 = c.select(qry5)

    qrr = "SELECT * FROM `subject` INNER JOIN `subject_staff_allocation` ON
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER JOIN
`timetable_change` ON `timetable_change`.`subject_id`=`subject`.`subject_id` INNER
JOIN `course` ON `course`.`course_id`=`subject`.`course_name`WHERE
`timetable_change`.date>CURDATE() AND `subject_staff_allocation`.`staff_id`='" +
str(
        session["l_id"]) + "'"
    db = Db()
    re = db.select(qrr)
    return render_template('teacher/Teacher veiw
Timeable.html',data=res,d1=d1,d2=d2,d3=d3,d4=d4,d5=d5,status="full",ll=len(re),re=re
)




@app.route('/teacher_view_allocate_timetable')
def teacher_view_allocate_timetable():
    c = Db()
    qry = "select * from course"
    res = c.select(qry)
    qq="SELECT * FROM `course` INNER JOIN `subject` ON
`subject`.`course_name`=`course`.`course_id` INNER JOIN `subject_staff_allocation`
ON `subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE
`subject_staff_allocation`.`staff_id`='"+str(session["l_id"])+"'"
    rr=c.select(qq)
    return render_template('teacher/Teacher view allocated
subject.html',data=res,data2=rr)

@app.route('/teacher_view_allocate_timetable_post',methods=['post'])
def teacher_view_allocate_timetable_post():
    course = request.form['select']
    sem = request.form['select2']
    c = Db()
    qry = "select * from course"
    res = c.select(qry)
    qq = "SELECT * FROM `course` INNER JOIN `subject` ON
`subject`.`course_name`=`course`.`course_id` INNER JOIN `subject_staff_allocation`
ON `subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE
`subject_staff_allocation`.`staff_id`='" + str(session["l_id"]) + "' and
course_id='"+course+"' and semester='"+sem+"'"
    rr = c.select(qq)
    return render_template('teacher/Teacher view allocated
subject.html',data=res,data2=rr)

@app.route('/teacher_view_complaint')
def teacher_view_complaint():
    qry="select * from `complaint` WHERE `staff_id`='"+str(session["l_id"])+"'"
    db=Db()
    res=db.select(qry)
    return render_template('teacher/Teacher view complaint.html',data=res)


@app.route('/teacher_view_profile')
def teacher_view_profile():
    qry="SELECT * FROM `staff` WHERE `login_id`='"+str(session["l_id"])+"'"
    db=Db()
    res=db.selectOne(qry)
    print(qry)
    return render_template('teacher/Teacher view profile.html',d=res)

@app.route('/teacher_view_status')
def teacher_VIEW_STATUS():
    qry="SELECT * FROM `leave_req` WHERE `staff_id`='"+str(session["l_id"])+"'"
    db=Db()
    res=db.select(qry)

    qq="SELECT COUNT(*) FROM `leave_req` WHERE `staff_id`='"+str(session["l_id"])+"' 
AND `status`='rejected'"
    ff=db.selectOne(qq)
    print(ff)
    qq1 = "SELECT COUNT(*) FROM `leave_req` WHERE `staff_id`='" + 
str(session["l_id"]) + "' AND `status`='pending'"
    ff1 = db.selectOne(qq1)

    qq11 = "SELECT COUNT(*) FROM `leave_req` WHERE `staff_id`='" + 
str(session["l_id"]) + "' "
    ff11 = db.selectOne(qq11)

    qq11 = "SELECT COUNT(*) FROM `leave_req` WHERE `staff_id`='" + 
str(session["l_id"]) + "'and status='accepted' and date<=curdate() "
    ff2 = db.selectOne(qq11)
    return render_template('teacher/TEACHER VIEW 
STATUS.html',data=res,rr=ff["COUNT(*)"],pr=ff1["COUNT(*)"],tr=ff11["COUNT(*)"],tlt=f
f2["COUNT(*)"])

@app.route('/admin_home')
def admin_home():
    return render_template('admin/admin_home.html')

@app.route('/t_home')
def t_home():
    return render_template('teacher/home.html')
@app.route('/index_page')
def index_page():
    return render_template('index.html')


@app.route("/teacher_view_class_timetable")
def teacher_view_class_timetable():
    db=Db()
    monday=[]
    tuesday=[]
    wednesday=[]
    thursday=[]
    friday=[]
    for i in range(1,6):
        qry="SELECT * FROM `subject` INNER JOIN `subject_staff_allocation` ON 
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER JOIN 
`time_table` ON `time_table`.`subject`=`subject`.`subject_id` WHERE 
`subject_staff_allocation`.`staff_id`='"+str(session["l_id"])+"' AND 
`time_table`.`day`='monday'AND `time_table`.`hour`='"+str(i)+"'"
        res=db.selectOne(qry)
        if res is not None:
            monday.append(res["subject_name"])
        else:
            monday.append("Free")
    for i in range(1,6):
        qry="SELECT * FROM `subject` INNER JOIN `subject_staff_allocation` ON 
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER JOIN 
`time_table` ON `time_table`.`subject`=`subject`.`subject_id` WHERE 
`subject_staff_allocation`.`staff_id`='"+str(session["l_id"])+"' AND 
`time_table`.`day`='tuesday'AND `time_table`.`hour`='"+str(i)+"'"
```

```
        res=db.selectOne(qry)
        if res is not None:
            tuesday.append(res["subject_name"])
        else:
            tuesday.append("Free")
    for i in range(1,6):
        qry="SELECT * FROM `subject` INNER JOIN `subject_staff_allocation` ON 
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER JOIN 
`time_table` ON `time_table`.`subject`=`subject`.`subject_id` WHERE 
`subject_staff_allocation`.`staff_id`='"+str(session["l_id"])+"' AND 
`time_table`.`day`='wednesday'AND `time_table`.`hour`='"+str(i)+"'"
        res=db.selectOne(qry)
        if res is not None:
            wednesday.append(res["subject_name"])
        else:
            wednesday.append("Free")
    for i in range(1,6):
        qry="SELECT * FROM `subject` INNER JOIN `subject_staff_allocation` ON 
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER JOIN 
`time_table` ON `time_table`.`subject`=`subject`.`subject_id` WHERE 
`subject_staff_allocation`.`staff_id`='"+str(session["l_id"])+"' AND 
`time_table`.`day`='thursday'AND `time_table`.`hour`='"+str(i)+"'"
        res=db.selectOne(qry)
        if res is not None:
            thursday.append(res["subject_name"])
        else:
            thursday.append("Free")
    for i in range(1,6):
        qry="SELECT * FROM `subject` INNER JOIN `subject_staff_allocation` ON 
`subject_staff_allocation`.`subject_id`=`subject`.`subject_id`INNER JOIN 
`time_table` ON `time_table`.`subject`=`subject`.`subject_id` WHERE 
`subject_staff_allocation`.`staff_id`='"+str(session["l_id"])+"' AND 
`time_table`.`day`='friday'AND `time_table`.`hour`='"+str(i)+"'"
        res=db.selectOne(qry)
        if res is not None:
            friday.append(res["subject_name"])
        else:
            friday.append("Free")
    qq = "SELECT * FROM `course` INNER JOIN `subject` ON 
`subject`.`course_name`=`course`.`course_id` INNER JOIN `subject_staff_allocation` 
ON `subject_staff_allocation`.`subject_id`=`subject`.`subject_id` WHERE 
`subject_staff_allocation`.`staff_id`='" + str(
        session["l_id"]) + "' and (semester='1' or semester='4' or semester='6')"
    rr = db.select(qq)
    print(rr)
    print(monday)
    print(tuesday)
    print(wednesday)
    print(thursday)
    print(friday)
    return 
render_template("teacher/Teacher_view_college_timetable.html",monday=monday,thursday
=thursday,tuesday=tuesday,wednesday=wednesday,friday=friday,data=rr)




if __name__ == '__main__':
    app.run(debug=True)
```

ANNEXURE-B

ABBREVIATION

- **IDE** - Integrated Development Environment
- **HTML -** Hypertext Mark-up Language
- **SQL** - Structured Query Language
- **UI** - User Interface
- **PK** - Primary Key
- **FK** - Foreign Key
- **SVGA –** Standard Video Graphics Array