**Zyphor**

*Project Report Submitted by*

**Shamjad Mazood Nazer**

**Reg. No.: AJC21MCA-2095**

*In Partial fulfillment for the Award of the Degree of*

**MASTER OF COMPUTER APPLICATIONS**

**(MCA TWO YEAR)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING**

**KANJIRAPPALLY**

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

# 2021-2023

## DEPARTMENT OF COMPUTER APPLICATIONS

### AMAL JYOTHI COLLEGE OF ENGINEERING

**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**ZYPHOR”** is the bona fide work of **SHAMJAD MAZOOD NAZER (Regno: AJC21MCA-2095)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2022-23.

**Mr. Rony Tom Ms. Meera Rose Mathew**

**Internal Guide Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**

**Head of the Department External Examiner**

**// CERTIFICATE ON PLAGIARISM CHECK**

**DECLARATION**

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

**Date: SHAMJAD MAZOOD NAZER**

**KANJIRAPPALLY Reg: AJC21MCA-2095**

# ACKNOWLEDGEMENT

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

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

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

SHAMJAD MAZOOD NAZER

# ABSTRACT

.

The Placement Management System Project is a web-based program that may be viewed both inside and outside the corporation with the right login credentials. This system can be used as an application by the college’s TPO to handle student information related to placement. This system is created with HTML, CSS, Bootstrap, JavaScript, as well as the backend is Python Programming Language and database as SQLite. The system is designed and created for the automation of campus drives and the compilation of reports, built for the training and placement department. With the right login, an online application can be accessed throughout the organization as well as outside of it.

**CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL. NO** | | **TOPIC** | **PAGE NO** | |
| **1** | | **INTRODUCTION** |  | |
| **1.1** | | **PROJECT OVERVIEW** |  | |
| **1.2** | | **PROJECT SPECIFICATION** |  | |
| **2** | | **SYSTEM STUDY** |  | |
| **2.1** | | **INTRODUCTION** |  | |
| **2.2** | | **EXISTING SYSTEM** |  | |
| **2.3** | | **DRAWBACKS OF EXISTING SYSTEM** |  | |
| **2.4** | | **PROPOSED SYSTEM** |  | |
| **2.5** | | **ADVANTAGES OF PROPOSED SYSTEM** |  | |
| **3** | | **REQUIREMENT ANALYSIS** |  | |
| **3.1** | | **FEASIBILITY STUDY** |  | |
| **3.1.1** | | **ECONOMICAL FEASIBILITY** |  | |
| **3.1.2** | | **TECHNICAL FEASIBILITY** |  | |
| **3.1.3** | | **BEHAVIORAL FEASIBILITY** |  | |
| **3.1.4** | | **FEASIBILITY STUDY QUESTIONNAIRE** |  | |
| **3.2** | | **SYSTEM SPECIFICATION** |  | |
| **3.2.1** | | **HARDWARE SPECIFICATION** |  | |
| **3.2.2** | | **SOFTWARE SPECIFICATION** |  | |
| **3.3** | | **SOFTWARE DESCRIPTION** |  | |
| **3.3.1** | | **PYTHON** |  | |
| **3.3.2** | | **SQLITE** |  | |
| **4** | | **SYSTEM DESIGN** |  | |
| **4.1** | | **INTRODUCTION** |  | |
| **4.2** | | **UML DIAGRAM** |  | |
| **4.2.1** | | **USE CASE DIAGRAM** |  | |
| **4.2.2** | | **SEQUENCE DIAGRAM** |  | |
| **4.2.3** | | **STATE CHART DIAGRAM** |  | |
| **4.2.4** | | **ACTIVITY DIAGRAM** |  | |
| **4.2.5** | | **CLASS DIAGRAM** |  | |
| **4.2.6** | | **OBJECT DIAGRAM** |  | |
| **4.2.7** | | **COMPONENT DIAGRAM** |  | |
| **4.2.8** | | **DEPLOYMENT DIAGRAM** |  | |
| **4.2.9** | | **COLLABORATION DIAGRAM** |  | |
| **4.3** | | **USER INTERFACE DESIGN USING FIGMA** |  | |
| **4.4** | | **DATABASE DESIGN** |  | |
| **5** | | **SYSTEM TESTING** |  | |
| **5.1** | | **INTRODUCTION** |  | |
| **5.2** | | **TEST PLAN** |  | |
| **5.2.1** | **UNIT TESTING** | |  |
| **5.2.2** | **INTEGRATION TESTING** | |  |
| **5.2.3** | **VALIDATION TESTING** | |  |
| **5.2.4** | **USER ACCEPTANCE TESTING** | |  |
| **5.2.5** | **AUTOMATION TESTING** | |  |
| **5.2.6** | **SELENIUM TESTING** | |  |
| **6** | **IMPLEMENTATION** | |  |
| **6.1** | **INTRODUCTION** | |  |
| **6.2** | **IMPLEMENTATION PROCEDURE** | |  |
| **6.2.1** | **USER TRAINING** | |  |
| **6.2.2** | **TRAINING ON APPLICATION SOFTWARE** | |  |
| **6.2.3** | **SYSTEM MAINTENANCE** | |  |
| **6.2.4** | **HOSTING** | |  |
| **7** | **CONCLUSION & FUTURE SCOPE** | |  |
| **7.1** | **CONCLUSION** | |  |
| **7.2** | **FUTURE SCOPE** | |  |
| **8** | **BIBLIOGRAPHY** | |  |
| **9** | **APPENDIX** | |  |
| **9.1** | **SAMPLE CODE** | |  |
| **9.2** | **SCREEN SHOTS** | |  |

## List of Abbreviation

HTML - Hypertext Markup Language

CSS - Cascade Style Sheet

SQL - Structured Query Language

UML - Unified Modeling Language

TPO - Training and Placement Officer

LPA - Lakhs Per Annum

IDE - Integrated Development Environment

CGPA - Cumulative Grade Point Average

# CHAPTER 1

# INTRODUCTION

### PROJECT OVERVIEW

The Zyphor (Placement Management System) Project is a web-based program that may be viewed both inside and outside the corporation with the right login credentials. This system can be used as an application by the college’s TPO to handle student information related to placement. This system is created with HTML, CSS, Bootstrap, JavaScript, as well as the Python Programming Language and SQLite database. The system is designed and created for the automation of campus drives and the compilation of reports, built for the training and placement department. With the right login, an online application can be accessed throughout our organization.

**System Users**

**Training and Placement Officer**

1. Login.
2. Manages the system.
3. Add the details of the placement drive.
4. View students enrolled.
5. Add mock test (Aiken Format).
6. View student’s test result.
7. Add training quiz.
8. View training quiz result.

**Student**

1. Register.
2. Login.
3. Payment.
4. Update details.
5. Apply for recruitment drives.
6. View status of the drive.
7. Attend the quiz.
8. View performance.
9. View the chances of getting placement.

### PROJECT SPECIFICATION

College placement cell is an important component of any college. The final year students appear for interviews offered by various companies invited by the placement officer. For this, there is a need for an online system where the placement officer can post the upcoming placement drives. Students willing to appear for these placement drives must be able to register with the system by a small amount of payment and get the message for any placement drive matching their qualification. This Python-Django web application, Zyphor Project helps the college placement cell’s functionalities fully in online, where the students will get only the drives which by their performance in academics. Students can attend the Mock test which is added by the TPO and the system will shows the performance of each students.

# CHAPTER 2

# SYSTEM STUDY

# 

### INTRODUCTION

System Study is a general term that refers to an orderly, structured process for identifying and solving problems. We call system Study process lifecycle methodology, since it relates to four significant phases in the lifecycle of all business information systems. The system is studied to the minute details and analyzed.

Analysis implies the process of breaking something into parts so that the whole may be understood. The definition of the system analysis includes not only the process of analysis but also that of synthesis, which implies the process of putting together to form a new whole. All activities associated with each life cycle phase must be performed, managed and documented. Hence we define system analysis as the performance, management, documentation of the activities related to the life cycle phases of a computer-based business system. In the study phase a detailed study of the project is made and clear a picture of the project is in mind. In the design phase the designing of the input, output and table designs are made. Development phase is where the physical designing of the input-output screens and coding of the system is done. System implementation actually implements the system by making necessary testing.

### EXISTING SYSTEM

### In order to elicit the requirements of the system and to identify the elements, input, outputs, subsystems and the procedures, the existing system had to be examined and analyzed in detail. This constitutes the system study. Records, slips, procedure, rules etc were examined thoroughly. The existing system was studied involving a complete cooperation from the employees who run the system at present.

**2.2.1 NATURAL SYSTEM STUDIED**

This system contains the following drawbacks. The procedure consists of different steps so immediate communication is not possible. Communication and file handling is not easy. The searching process for the existing system is not possible. Frequent updating is not possible.

**2.2.2 DESIGNED SYSTEM STUDIED**

This system keeps track of student information related to placement. It enhances the current system. It offers the capability of storing the student’s information, decreasing the amount of human effort required. It will save time and energy that would otherwise be spent on creating reports and gathering data.

### DRAWBACKS OF EXISTING SYSTEM

* Need a large number of different registers.
* Chance of losing information.
* Need a large volume of storage area.
* Reports could not be prepared in time.
* Use and maintenance of paper files, registers and other written matter was very difficult.
* Security of the system depended on the persons dealing with it.
* Make performance analysis of each students are difficult.

### PROPOSED SYSTEM

The web application entitled “Zyphor Project” deals with student’s placement/recruitment process. The students can upload their academic details, extracurricular activities and achievements till the due date of the drive by permitted date. There are two types of users in this system-admin (TPO), and students. The admin can view the reports and the student's status of academic performance and can filter/sort them in the system for mailing purposes.

### ADVANTAGES OF PROPOSED SYSTEM

* Interactive and user friendly environment.
* Information can be retrieved from anywhere and at any time.
* Providing accurate and timely information.
* Mock test provided for the students.
* Aptitude training for students.

# CHAPTER 3

# REQUIREMENT ANALYSIS

## FEASIBILITY STUDY

A feasibility study is a test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of resources. The objective of feasibility study is acquiring a sense of the scope of the system.

The feasibility of a project can be ascertained in terms of technical factors, economic factors, or both. A feasibility study is documented with a report showing all the ramifications of the project. It is very important to evaluate the feasibility study and risk analysis is related in many ways. If the project study is great, the feasibility of producing quality software is reduced.

The key factors considered during the feasibility study are:

1. Economic Feasibility
2. Behavioral Feasibility
3. Technical Feasibility
4. Legal Feasibility

### Economic Feasibility

Economic analysis is the most commonly used method for evaluating effectiveness of a system. Cost-benefit analysis is the most important assessment of economic justification of the project. Cost –benefit analysis delineates the cost for project development and weighs them against tangible and intangible benefits of a system. This type of analysis varies with the characteristics of the system to be developed, the relative site of the project, and the expected return on investment. Benefits of a new system are always determined relative to the existing mode of operation.

Economic feasibility deals with the economic impact faced by the organization to implement the new system. Not only the cost of hardware, software etc. are considered but also the form of reduced costs. The project, installed, will certainly be beneficial since there will be a reduction in manual work, increase in speed of work.

The analysis raises financial and economic questions during the preliminary investigation to estimate the following:

1. The cost to conduct a full systems investigation.
2. The cost of hardware and software for the class of application of the project being considered.

To be judged feasible, a proposal for the specific project must pass all these tests, otherwise it is not considered as a feasible project. I gathered the details regarding the financial aspects incorporated in the system to make it cost efficient.

### Technical Feasibility

There are a number of technical issues, which are generally raised during the feasibility stage of the investigation. A study of function, performance and constraints gave me the ability to achieve acceptable system. The software required for this system is:

1. Python
2. SQLite

### Behavioral Feasibility

### Proposed projects are beneficial only if they can be turned into information systems that will meet the operating requirements of the organization. This test of feasibility asks if the system will work when it is developed and satisfies all the operational conditions. It was the most difficult task for me, but met efficiently. As this package is found to be feasible technically, economically and functionally, the system is judged feasible. Viewing the collected information, recommendation and justification, conclusions are made of the proposed system. Hence decision is taken to go on with the project.

### Feasibility Study Questionnaire

1. Who are the users of the system?
2. TPO, Student, Class Teachers.
3. What is the role of TPO?

A) Can filter and sort the students as department, marks, backlogs, CGPA, gender, technology known etc.and can send SMS/mail to the filtered candidate.

1. What do the students do in the system?

A) Students can register to the placement cell via registration form and completing the payment, they can update their profile and can apply to the placement drives.

1. What is the role of the class teacher in this system?

A) Here, semester wise published marks of the students will be updated by the class teacher.

1. What is the role of HOD?

A) They can view the performance of their own students in the department.

1. Who will alert the students about the job?

A) The TPO is responsible for that. JD (Job Description) is added/updated by the TPO and all the students who registered within the placement cell will be informed via mail.

1. What if a student got already placed?

A) They cannot receive any further updates about any new drives.

1. What are the details should we collect from the student while registering?

A) When a student is registering to the placement cell, we collect each and every details of the students about their academics, personal details, and general details such as PAN number, passport details, blood group etc.

1. Is there anything that can be added to a student's profile?

A) They can update their details such as certifications, projects, internships, add-on courses etc.

1. Additional requirements?
2. TPO can sort the students based on their academic performance, and send the mail to those students about the drive.
3. Are there any measures in place to assist students in preparing for job interviews and improving their soft skills?
4. Make a training platform for the students which they can perform aptitude assessment and can analyze their performance by themselves.
5. How do you handle situations where students are not placed or do not receive satisfactory job offers?
6. Majority of the companies provide offer letter to the students who got placed from our college and a few companies make some lagging for onboarding. For those students, we provide them to attend the upcoming drives.
7. Based on your experience and observations, do you believe there is a need for improvement or enhancement in the existing placement cell system? If yes, what areas would you prioritize for improvement?
8. Yes, currently we don’t have a provision for giving the mock test for students. So we can give the students mock tests as a preparation before attending any drives. By this, the students can assess themselves and make necessary preparations for them.
9. Should there be any time restrictions or deadlines for completing the mock tests?
10. Yes, like the actual online aptitude test we can do the same within our system too.
11. Would you like to include explanations or solutions for the questions after the mock test?
12. Of course, at the end of the quiz students can view their performance of the quiz along with the question and its answers.

## 3.2 SYSTEM SPECIFICATION

## INTRODUCTION

## Requirement analysis involves studying the current system to find out how it works and here improvements could be made. A clear idea about the existing system is a must for making improvements where it is required. Proper planning and collection of data serves the purpose. The popularity of this document is to describe all the requirements for the popularity of the software Department Management. In our busy lives, we cannot go to each company or office. This problem will manage this site. It will help to avoid corrections and missing data.

### Hardware Specification

Processor - Dual Core

RAM - 2 GB RAM

Hard disk - 120 GB

### Software Specification

Front End - HTML, CSS, Bootstrap

Backend - SQLite

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, JQuery, Python-Django, CSS

## SOFTWARE DESCRIPTION

### INTRODUCTION TO PYTHON

### Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamic typed language. For example, x=10, here x can be anything such as String, int etc.

### Features in Python

There are many features in Python, some of which are discussed below –

1. Easy to code:

Python is a high-level programming language. Python is very easy to learn language as compared to other language like C,Cc#, JavaScript, Java etc. It is very easy to code in python language and anybody can learn python basic in a few hours or days. It is also a developer-friendly language.

1. Free and Open Source:

Python language is freely available at the official website and you can download it. Since, it is open-source; this means that source code is also available to the public. So you can download it, use it as well as share it.

1. Object-Oriented Language:

One of the key features of python is Object-Oriented programming. Python supports object oriented language and concepts of classes, object encapsulation etc.

1. High-Level Language:

Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

1. Python is Portable Language

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, UNIX and Mac then we do not need to change it, we can run this code on any platform.

1. Large Standard Library:

Python has a large standard library which provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in python such as regular expressions, unit-testing, web browsers etc.

### SQLite

### This SQLite tutorial teaches you everything you need to know to start using SQLite effectively. In this tutorial, you will learn SQLite step by step through extensive hands-on practices.

### This SQLite tutorial is designed for developers who want to use SQLite as the back-end database or to use SQLite to manage structured data in applications including desktop, web, and mobile apps.

### SQLite is an open-source, zero-configuration, self-contained, stand-alone, transaction relational database engine designed to be embedded into an application.

# CHAPTER 4

# SYSTEM DESIGN

* 1. **INTRODUCTION**

The most creative and challenging phase of the system development is system design, is a solution to how to approach the creation of the proposed system. It refers to the technical specification that will be applied. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Design goes through the logical and physical stages of development. At an early stage in designing a new system, the system analyst must have a clear understanding of the objectives, which the design is aiming to fulfill. The first step is to determine how the output is to be produced and in what format. Second input data and master files (database) have to be designed to meet the requirements of the proposed output. The operational (processing) phases are handled through program construction and testing.

The system design involves first logical design and then physical construction of the system. The logical design describes structure and characteristics of features, like the outputs, inputs, databases and procedures. The physical construction which follows the logical design produces actual program software files and the working system.

System design sits at the technical kernel of software engineering and is applied regardless of the software process model that is used. Beginning once software requirements have been analyzed and specified, software design is the first technical activity that is used to build and verify the software. Each activity (designing, coding and testing) transforms information in a manner that ultimately results in validated computer software.

## UML DIAGRAM

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

UML include the following diagrams:

* Class diagram
* Object diagram
* Use case diagram
* Sequence diagram
* Activity diagram
* State chart diagram
* Deployment diagram
* Component diagram

## USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between system components. An approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" here refers to a thing that is being created or run, like a website for mail-order product sales and services. UML (Unified Modeling Language), a standard language for the modeling of real-world objects and systems, uses use case diagrams.

The planning of general requirements, the validation of a hardware design, the testing and debugging of a software product in development, the creation of an online help reference, or the completion of a job focused on customer support are all examples of system objectives. For instance, use cases in a product sales context can involve customer service, item ordering, catalog updating, and payment processing. There are four elements in a use case diagram.

* The boundary, which defines the system of interest in relation to the world around it.
* The actors, usually individuals involved with the system defined according to their roles.
* The use cases, which are the specific roles are played by the actors within and around the system.
* The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram.

## 4.2.2 SEQUENCE DIAGRAM

A sequence diagram essentially shows how things interact with one another sequentially, or the order in which these interactions occur. A sequence diagram can also be referred to as event diagrams or event scenarios. Sequence diagrams show the actions taken by the components of a system in chronological order. Businesspeople and software engineers frequently use these diagrams to record and comprehend the requirements for new and current systems.

Sequence Diagram Notations –

1. **Actors** – In a UML diagram, an actor represents a particular kind of role that interacts with the system and its objects. An actor is always beyond the purview of the system that we want to use the UML diagram to represent. We employ actors to portray a variety of roles, including those of human users and other outside subjects. In a UML diagram, an actor is represented using a stick person notation. In a sequence diagram, there might be several actors.
2. **Lifelines** – A lifeline is a named element in a sequence diagram that represents an individual participant. So, in a sequence diagram, each incident is represented by a lifeline. A sequence diagram's lifeline elements are at the top.
3. **Messages** – Messages are used to show how objects communicate with one another. The messages are displayed on the lifeline in chronological sequence. Arrows are how messages are represented. A sequence diagram's main components are lifelines and messages.

Messages can be broadly classified into the following categories:

* Synchronous messages
* Asynchronous Messages
* Create message
* Delete Message
* Self-Message
* Reply Message
* Found Message
* Lost Message

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

**Uses of sequence diagrams –**

* Used to model and visualize the logic behind a sophisticated function, operation or procedure.
* They are also used to show details of UML use case diagrams.
* Used to understand the detailed functionality of current or future systems.
* Visualize how messages and tasks move between objects or components in a system.

## 4.2.3 State Chart Diagram

Explanation, Diagram

## 4.2.4 Activity Diagram

Another crucial UML diagram for describing the system's dynamic elements is the activity diagram. An activity diagram is essentially a flowchart that shows how one activity leads to another. The action might be referred to as a system operation. One operation leads to the next in the control flow. This flow may be parallel, contemporaneous, or branched. Activity diagrams use many features, such as fork, join, etc., to cope with all types of flow control. An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

Explanation, Diagram

## Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling.

Explanation, Diagram

## Object Diagram

Since class diagrams are the source of object diagrams, class diagrams are a prerequisite for object diagrams. An instance of a class diagram is represented by an object diagram. Class and object diagrams both use the same fundamental ideas. The static view of a system is also represented by object diagrams, but this static view represents a momentary snapshot of the system. To represent a group of items and their connections as an instance, object diagrams are employed.

Explanation, Diagram

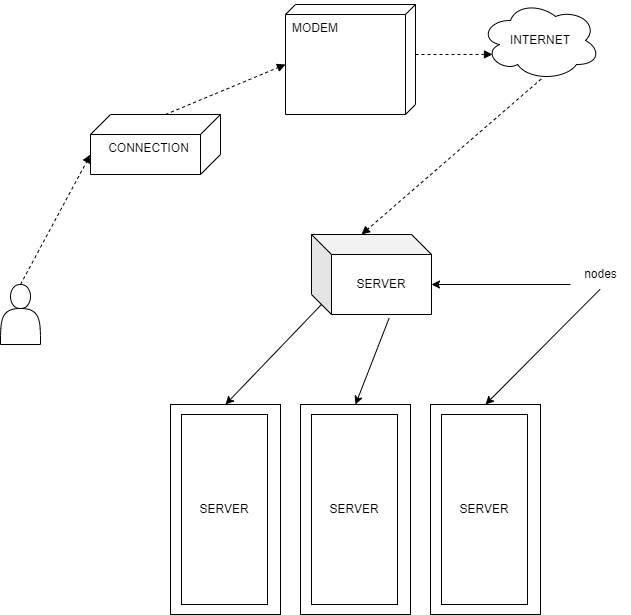
## Component Diagram

A component diagram depicts how components are wired together to form larger components of software systems. They are used to illustrate the structure of arbitrarily complex systems

Explanation, Diagram

**4.2.8 Deployment Diagram**

An execution architecture of a system, containing nodes like hardware or software execution environments, and the middleware linking them, is shown in a deployment diagram, a form of UML diagram. Typically, deployment diagrams are used to represent the actual hardware and software of a system. By using it, you can comprehend how the hardware will physically deliver the system. In contrast to other UML diagram types, which primarily depict the logical components of a system, deployment diagrams assist describe the hardware structure of a system.

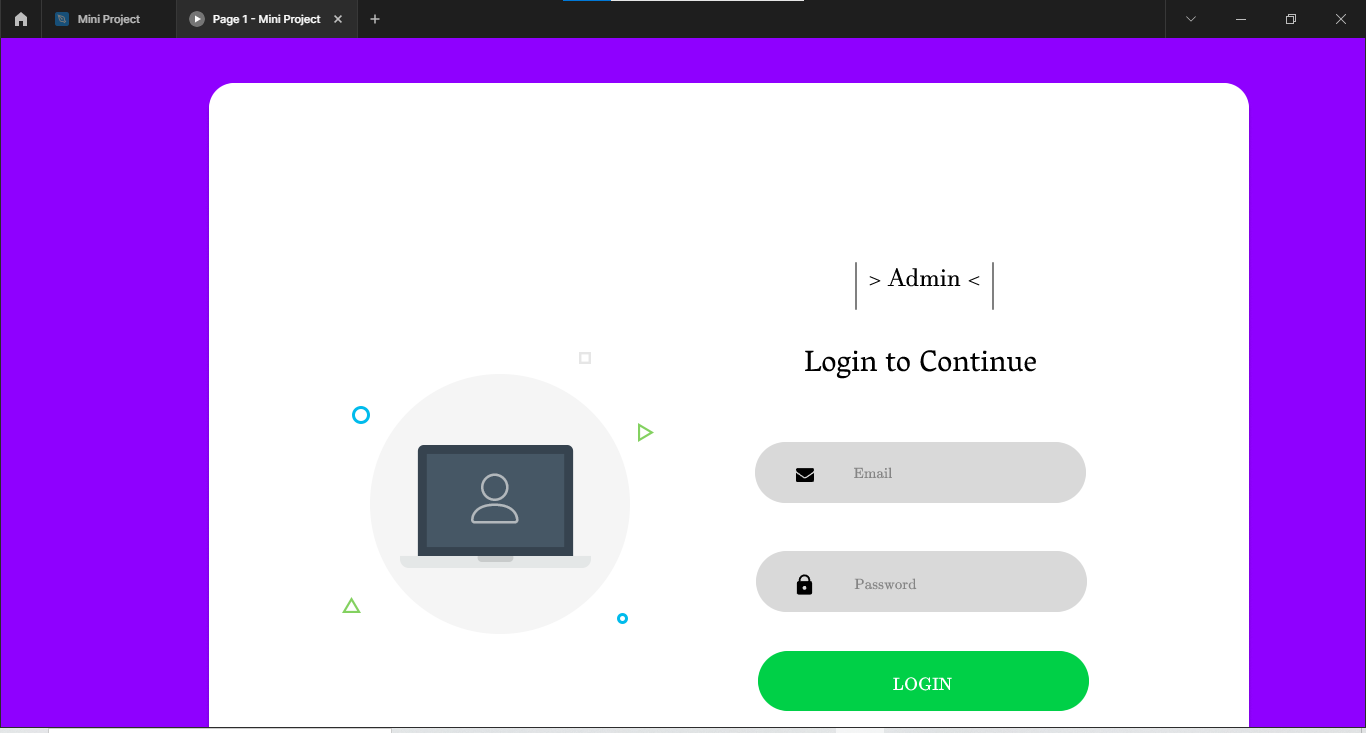


**4.2.9 Collaboration Diagram**

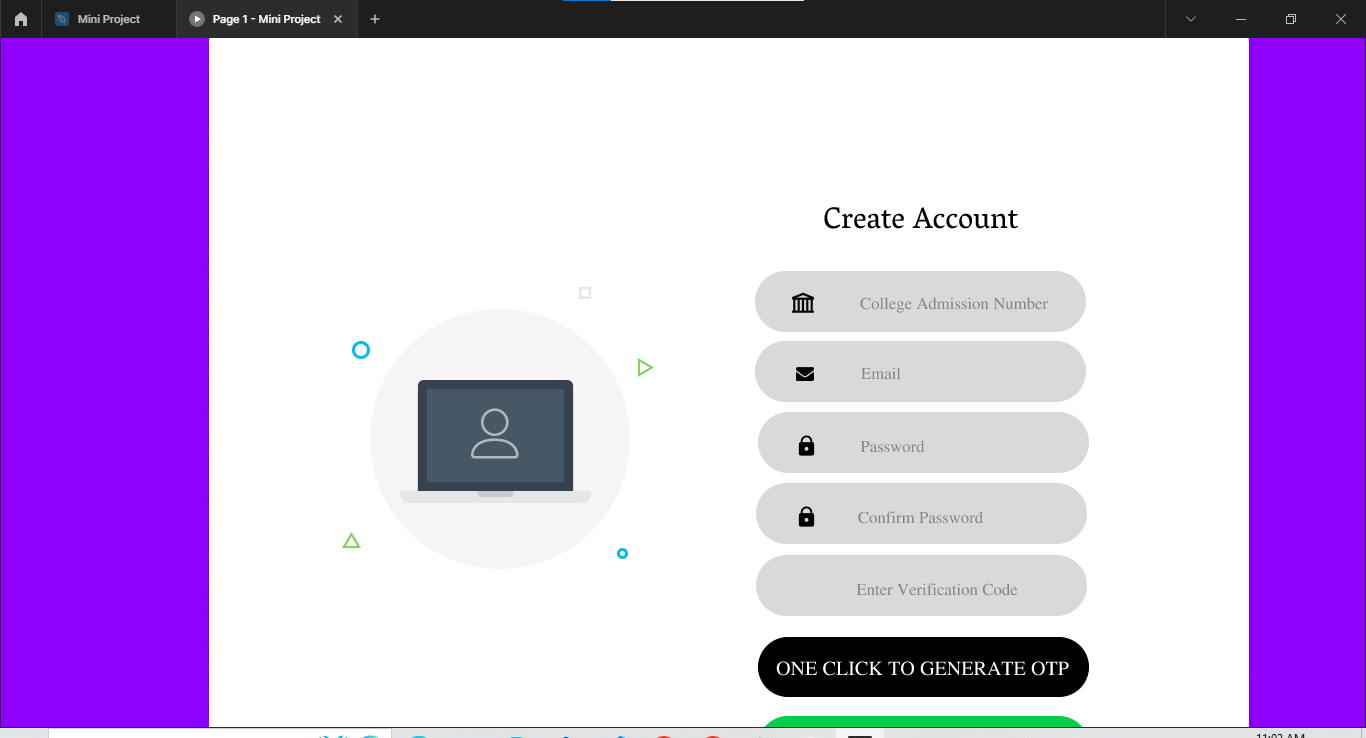
Explanation, Diagram

## 4.3 USER INTERFACE DESIGN USING FIGMA

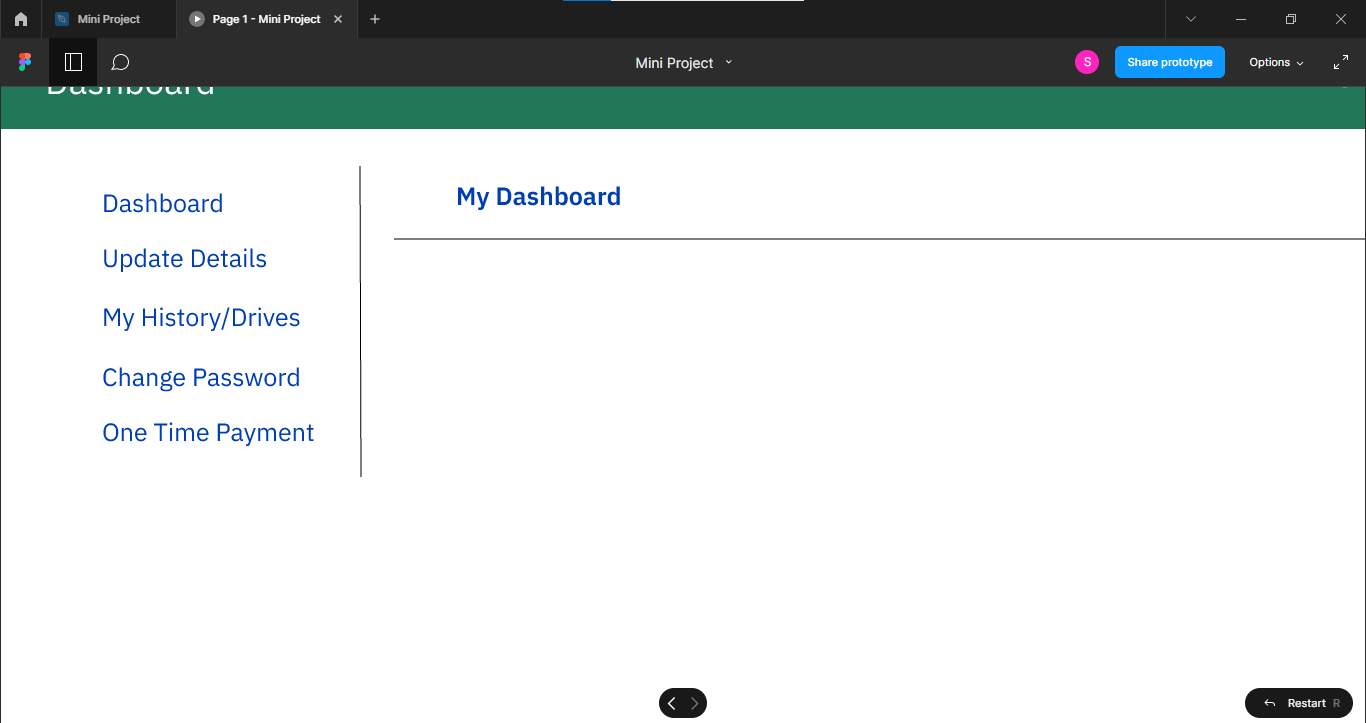
**Form Name: Login**



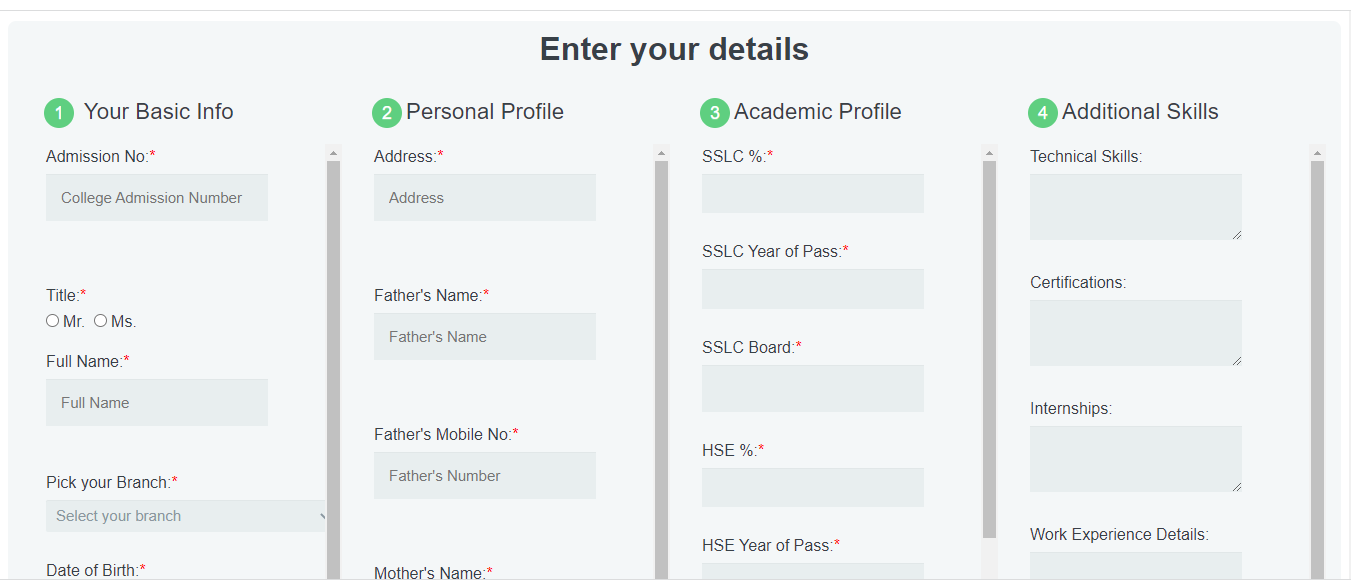
**Form Name: Registration Form**



**Form Name: Student Dashboard**



**Form Name: Student Details**



**Form Name: Update Details**

## DATABASE DESIGN

System design sits at the technical kernel of software engineering and is applied regardless of the software process model that is used. Beginning once software requirements have been analyzed and specified, software design is the first technical activity that is used to build and verify the software. Each activity (designing, coding and testing) transforms information in a manner that ultimately results in validated computer software.

### 4.4.1 Relational Database Management System (RDBMS)

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected. The database design is a two-level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS. In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

* Data Integrity
* Data independence

### 4.4.2 Normalization

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a tale represents a set of related values.

**Relations, Domains & Attributes**

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values. Every value in a relation is atomic, that is not decomposable.

* Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
* Entity Integrity enforces that no Primary Key can have null values.
* Referential Integrity enforces that no Primary Key can have null values.
* Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other keys are Super Key and Candidate Keys.

**Normalization**

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is the formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal forms in data modeling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies records from a different table. All the tables have been normalized up to the third normal form.

As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups of data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These include:

* Normalize the data.
* Choose proper names for the tables and columns.
* Choose the proper name for the data.

**First Normal Form**

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words, 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be done by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

**Second Normal Form**

According to the Second Normal Form, for relations where the primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and set up a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attribute of the relation is fully dependent on its primary key alone.

**Third Normal Form**

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relations that include the non-key attributes that functionally determine other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and moreover the non key attributes of the relation should not be dependent on another non-key attribute.

### 4.4.3 Sanitization

Sanitizing data means removing any illegal character from the data. Sanitizing user input is one of the most common tasks in a web application. To make this task easier Python provides native filter extension that you can use to sanitize the data such as e-mail addresses, URLs, IP addresses, etc.

Python filters are used to sanitize and validate external input. The Python filter extension has many of the functions needed for checking user input, and is designed to do data sanitization easier and quicker. This function, when using the flag in the example, is making sure that the code removes all characters except letters, digits and the following characters !#$%&’\*+-=?\_`{|}~@.[] . Many web applications receive external input. External input/data can be:

* User input from a form
* Cookies
* Web services data
* Server Variables
* Database query results

**4.4.4 Indexing**

The index stores the value of a specific field or set of fields, ordered by the value of the field. The ordering of the index entries supports efficient equality matches and range-based query operations. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records. Indexes support the efficient execution of queries in Python. An "index" can improve the speed of operation in a table. SQLite automatically creates an index for primary key, foreign key, and unique constraints. In addition, you may want to create "indexes" for other columns that are frequently used in joins or search conditions. The user cannot see indexes. You must have used a "CREATE INDEX" statement to create an index for one or more columns of a table. To create an index, write the table name and column names after the "on" clause. You can also use "UNIQUE" keywords to specify that an "index" has only unique values. You can also specify "ASC" and "DESC" keywords with a column name to indicate whether you want the "index" stored in ascending or descending order. If you do not specify "asc" or "desc", then "asc" is the default same as the "order by" keyword (which is also able to sort columns in "asc" or "desc" order.

### TABLE DESIGN

1. **Table Name: StudentReg**

Primary key: **StudentReg\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | StudentReg\_id | int | Primary Key | Primary key of the Table |
| 2 | StudentReg\_Admission\_number | int | Unique | Admission number of the Student |
| 3 | StudentReg\_Email | varchar(90) | Unique | Personal mail of the Student |
| 4 | StudentReg\_First\_name | varchar(20) | Not null | First name of the Student |
| 5 | StudentReg\_Middle\_name | varchar(20) | Null | Middle name of the Student |
| 6 | StudentReg\_Last\_name | varchar(20) | Not null | Last name of the Student |
| 7 | StudentReg\_Password | varchar(25) | Not null | Password of the Student |

1. **Table Name: Tpo**

Primary key: **tpo\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | tpo\_id | int | Primary Key | Primary key of the Table |
| 2 | tpo\_name | varchar(50) | Not null | Name of the TPO |
| 3 | tpo\_mail | varchar(50) | Unique | Mail of the TPO |
| 4 | tpo\_password | varchar(20) | Not null | Password of the TPO |

1. **Table Name: Quiz**

Primary key: **quiz\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | Quiz\_id | Int | Primary key | Primary key of the Table |
| 2 | Quiz\_title | Varchar(20) | Not Null | Title name of the quiz |

1. **Table Name: Payment**

Primary key: **Payment\_id**

Foreign key:  **StudentReg\_id** references table **StudentReg**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | Payment\_id | Int | Primary key | Primary key of the Table |
| 2 | StudentReg\_id | Int | Foreign key | id from the table StudentReg |
| 3 | Payment\_Token | Varchar(50) | Not null | Mail of the TPO |
| 4 | Payment\_on | Date | Date | When did the payment done |
| 5 | Payment\_status | Varchar(20) | Not null | Password of the TPO |

1. **Table Name: AikenFile**

Primary key: AikenFile\_id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | Aikenfile\_id | Int | Primary key | Primary key of the Table |
| 2 | Aikenfile\_name | Varchar(50) | Not Null | Name of the Quiz |
| 3 | Aikenfile\_uploaded\_on | Date | Date | Date which the file is been uploaded |
| 4 | Aikenfile\_file | File | Not Null | Text file written with Aiken format |
| 5 | Aikenfile\_time | Int | Not Null | Total time that allotted to attend the quiz. |
| 6 | Aikenfile\_start\_date | Date | Date | Date when the quiz will start |
| 7 | Aikenfile\_end\_date | Date | Date | Date when the quiz will end |

1. **Table Name: Question**

Primary key: **question\_id**

Foreign key: **quiz\_id** references table **Quiz**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | question\_id | Int | Primary key | Primary key of the Table |
| 2 | Quiz\_id | Int | Foreign key | id from the table Quiz |

1. **Table Name: Answer**

Primary key: **answer\_id**

Foreign key: **question\_id** references table **Question**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | answer\_id | Int | Primary key | Primary key of the Table |
| 2 | question\_id | Int | Foreign key | id from the table Question |
| 3 | answer\_text | Varchar(255) | Not Null | Answer options for the questions on table Question |
| 4 | is\_correct | Boolean | Not Null | True only for the correct option |

1. **Table Name: AikenResult**

Primary key: **result\_id**

Foreign key: **quiz\_id** references table **Quiz**

Foreign key: **user** references table **StudentReg**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | result\_id | Int | Primary key | Primary key of the Table |
| 2 | quiz\_id | Int | Foreign key | id from the table Quiz |
| 3 | user\_id | Int | Foreign key | id from the table StudentReg |
| 4 | score | Int | Not Null | Score secured on aptitude test |
| 5 | time | Time | Not Null | Time taken to complete the test |
| 6 | correct | Int | Not Null | Count of correct answers |
| 7 | wrong | Int | Not Null | Count of wrong answers |
| 8 | percent | Int | Not Null | Percentage scored on the test |
| 9 | total\_question | Int | Not Null | Total number of questions attended from the test |
| 10 | quiz\_taken\_on | Date | Not Null | Date of the test is attended |

1. **Table Name: TrainingQuiz**

Primary key: **training\_quiz\_id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | training\_quiz\_id | Int | Primary key | Primary key of the Table |
| 2 | question | Varchar(255) | Not Null | Question text for the training quiz |
| 3 | op1 | Varchar(255) | Not Null | Option1 of the Question |
| 4 | op2 | Varchar(255) | Not Null | Option2 of the Question |
| 5 | op3 | Varchar(255) | Not Null | Option3 of the Question |
| 6 | op4 | Varchar(255) | Not Null | Option4 of the Question |
| 7 | ans | Varchar(7) | Not Null | Correct answer’s option name |

1. **Table Name: LanguagesKnown**

Primary key: **languages\_known\_id**

Foreign key: **studentDetails\_id** references table **StudentDetails**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | language\_known\_id | Integer | Primary key | Primary key of the table |
| 2 | studentDetails\_id | Integer | Foreign key | id from the table StudentDetails |
| 3 | language\_name | Varchar(20) | Not null | Name of the language |
| 4 | Language\_speak | Boolean | Null | Select if the language can speak |
| 5 | Language\_write | Boolean | Null | Select if the language can write |
| 6 | Language\_read | Boolean | Null | Select if the language can read |

1. **Table Name: StudentDetails**

Primary key: **studentDetails\_id**

Foreign key: S**tudentReg\_id** references table **StudentReg**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | studentDetails\_id | Integer | Primary key | Primary key of the table |
| 2 | StudentReg\_id | Integer | Foreign key | id from the table StudentReg |
| 3 | StudentDetails\_Branch | Varchar(25) | Not null | Which stream is student belongs |
| 4 | StudentDetails\_Date of birth | Date | Not null | Date of birth of student |
| 6 | StudentDetails\_Gender | Varchar(6) | Not null | Gender of student |
| 7 | StudentDetails\_Mobile number | Integer | Not null | Primary phone number of the Student |
| 8 | StudentDetails\_Alternative  Number | Integer | Not null | Alternative Mobile number |
| 9 | StudentDetails\_College mail | Varchar(90) | Not null | college mail of the Student |
| 10 | StudentDetails\_Father’s name | Varchar(30) | Not null | Father’s Name of the Student |
| 11 | StudentDetails\_Father’s\_number | Integer | Not null | Father’s Mobile Number |
| 12 | StudentDetails\_Mother’s\_name | Varchar(30) | Not null | Mother’s Name of the Student |
| 13 | StudentDetails\_Mother’s\_number | Integer | Not null | Mother’s Phone Number |
| 14 | StudentDetails\_house\_name | Varchar(50) | Not null | House name of the Student |
| 15 | StudentDetails\_post\_office | Varchar(50) | Not null | Post office name of the Student |
| 16 | StudentDetails\_city | Varchar(50) | Not null | Name of the city |
| 17 | StudentDetails\_pincode | Integer | Not null | Pincode of the post office |
| 18 | StudentDetails\_district | Varchar(25) | Not null | District name of the Student |
| 19 | StudentDetails\_Nationality | Varchar(25) | Not null | Nationality of the student |
| 20 | StudentDetails\_Plans\_after\_graduation | Varchar(25) | Not null | Higher studies, Start-ups, Career |
| 21 | StudentDetails\_Sslc\_per | Integer | Not null | Percentage of the 10th board exam |
| 22 | StudentDetails\_Sslc\_year\_of\_pass | Integer | Not null | Year of pass of 10th |
| 23 | StudentDetails\_Sslc\_board | Varchar(20) | Not null | Board name of 10th |
| 24 | StudentDetails\_HSE\_per | Integer | Not null | Percentage got from higher secondary exam |
| 25 | StudentDetails\_HSE\_year\_of\_pass | Integer | Not null | Year of pass of HSE |
| 26 | StudentDetails\_HSE\_board | Varchar(20) | Not null | Board name of HSE |
| 27 | StudentDetails\_UG\_program | Varchar(30) | Not null | Name of the UG program |
| 28 | StudentDetails\_UG\_per | Integer | Not null | Total percentage scored on UG |
| 29 | StudentDetails\_UG\_CGPA | Integer | Not null | Total CGPA scored on UG |
| 30 | StudentDetails\_UG\_year\_of\_pass | Integer | Not null | Year of pass of UG course |
| 31 | StudentDetails\_UG\_college\_name | Varchar(50) | Not null | Name of College that UG studied |
| 32 | StudentDetails\_UG\_university | Varchar(50) | Not null | Name of UG University |
| 33 | StudentDetails\_Entrance\_rank | Integer | Not null | Entrance Rank scored |
| 34 | StudentDetails\_MCA\_aggr\_CGPA | Integer | Not null | Aggregate CGPA of MCA |
| 35 | StudentDetails\_Active\_arrears | Integer | Not null | Count of active arrears |
| 36 | StudentDetails\_History\_arrears | Integer | Not null | Count of arrears history |
| 37 | StudentDetails\_Exam\_not\_attended | Integer | Not null | Count of exams not attended |
| 38 | StudentDetails\_University | Varchar(50) | Not null | Name of current University |
| 39 | StudentDetails\_Technical\_skills1 | Varchar(255) | Not null | Additional Skill that students have |
| 40 | StudentDetails\_Technical\_skills2 | Varchar(255) | Not null | Additional Skill that students have |
| 41 | StudentDetails\_Technical\_skills3 | Varchar(255) | Not null | Additional Skill that students have |
| 42 | StudentDetails\_Technical\_skills4 | Varchar(255) | Not null | Additional Skill that students have |
| 43 | StudentDetails\_Technical\_skills5 | Varchar(255) | Not null | Additional Skill that students have |
| 44 | StudentDetails\_Technical\_skills6 | Varchar(255) | Not null | Additional Skill that students have |
| 45 | StudentDetails\_Technical\_skills7 | Varchar(255) | Not null | Additional Skill that students have |
| 46 | StudentDetails\_Technical\_skills8 | Varchar(255) | Not null | Additional Skill that students have |
| 47 | StudentDetails\_Github\_profile | Varchar(255) | Not null | URL of GitHub Profile |
| 48 | StudentDetails\_Linkedin\_profile | Varchar(255) | Not null | URL of LinkedIn Profile |
| 49 | StudentDetails\_Upload\_photo | File | Not null | Profile photo of the Student |

1. **Table Name: Certificates**

Primary key: **certificates\_id**

Foreign key: **studentDetails\_id** references table **StudentDetails**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Fieldname** | **Datatype (Size)** | **Key Constraints** | **Description of the Field** |
| 1 | certificates\_id | Integer | Primary key | Primary key of the table |
| 2 | studentDetails\_id | Integer | Foreign key | id from the table StudentDetails |
| 3 | certificate\_name | Varchar(50) | Not null | Name of the certificate |
| 4 | start\_date | Date | Not null | Start date of the certificate achieved |
| 5 | end\_date | Date | Not null | End date of the certificate achieved |
| 6 | certified\_date | Date | Not null | Date of the certificate issued |
| 7 | certificate\_file | File | Not null | Certificate file |

# CHAPTER 5

# SYSTEM TESTING

* 1. **INTRODUCTION**

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

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

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

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

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

1. For accuracy
2. For effective implementation
3. Regarding computational complexity

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

## TEST PLAN

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

The levels of testing include:

1. Unit testing

2. Integration Testing

3. Data validation Testing

4. Output Testing

### Unit Testing

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. The level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enter and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested. Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling paths must be put up to cleanly reroute or halt work when an error does arise. The final step of unit testing is boundary testing. Software frequently fails at its limits.

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

### Integration Testing

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

### Validation Testing or System Testing

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing.

The functional requirements of the software are the main emphasis of the black box testing approach. For example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every program requirement.

The following sorts of problems are targeted by black box testing: erroneous or missing functions, interface errors, data structure or external data access errors, performance errors, initialization errors, and termination errors.

### Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with the prospective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

1. Input Screen Designs
2. Output Screen Designs

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted.

* + 1. **Automation Testing**

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

* + 1. **Selenium Testing**

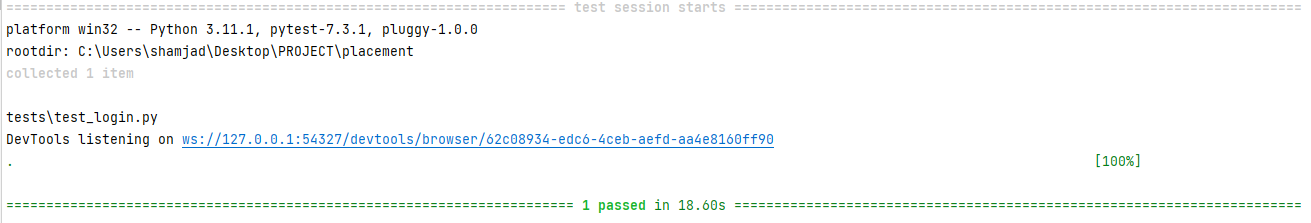
An open-source program called Selenium automates web browsers. It offers a single interface that enables you to create test scripts in a number of different programming languages, including Ruby, Java, NodeJS, PHP, Perl, Python, and C #. Web application testing for cross browser compatibility is automated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

**Test Case 1: Student Login**

**Code**



**Screenshot**

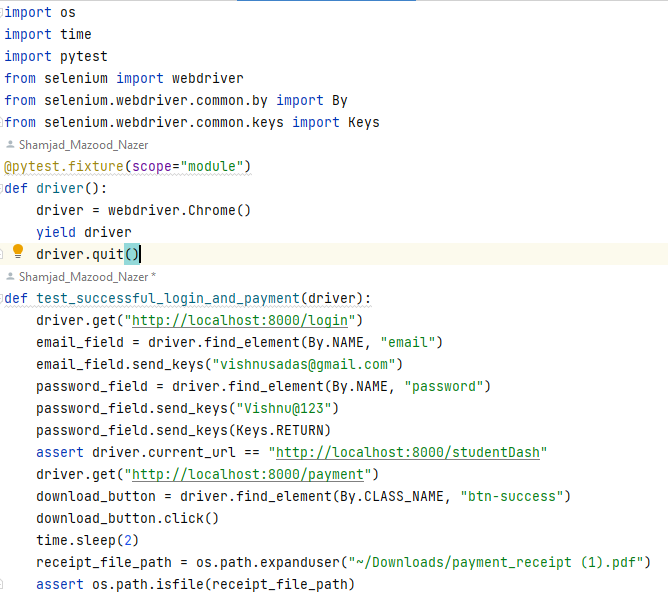


**Test Report**

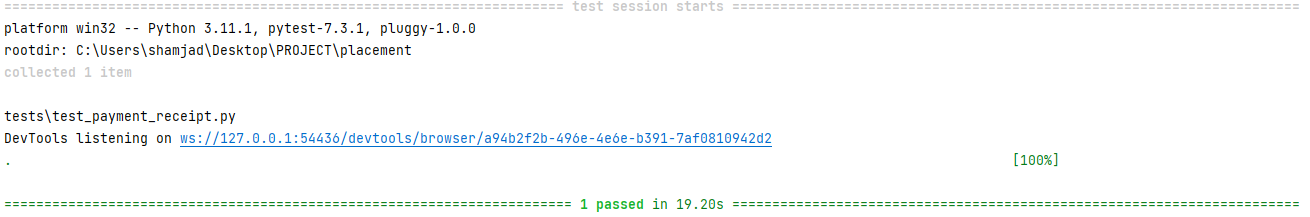
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 1** | | | | | |
| **Project Name: Zyphor** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID:** Test\_1 | | | **Test Designed By:** Shamjad Mazood Nazer | | |
| **Test Priority(Low/Medium/High):** High | | | **Test Designed Date:** 10-05-2023 | | |
| **Module Name**: Login | | | **Test Executed By :** Mr. Rony Tom | | |
| **Test Title :** Student Login | | | **Test Execution Date:** 10-05-2023 | | |
| **Description:** verify login with a valid email and password | | |  | | |
| **Pre-Condition:** User has valid email and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Accessing the login page |  | Login page is to be visible | Login page appeared | Pass |
| 2 | Give a real email | Email: [shamjad.nazar.20@gmail.com](mailto:shamjad.nazar.20@gmail.com) | User should be able to login | User logged in and redirected to the dashboard | Pass |
| 3 | Provide a valid password | Password: Admin@123 |
| 4 | Select the login button |  |
| **Post-Condition:** The user has successfully authenticated with the database and logged into the account. The database contains information about the account session | | | | | |

**Test Case 2:** Payment Receipt

**Code**

****

**Screenshot**

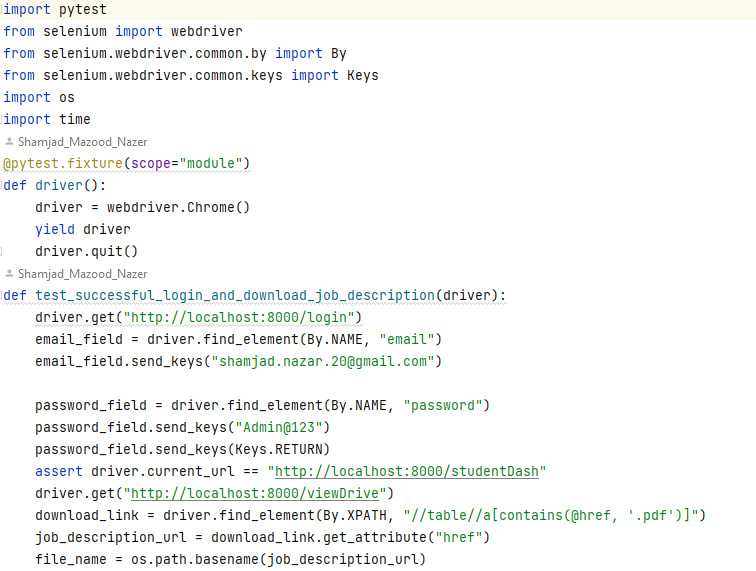
****

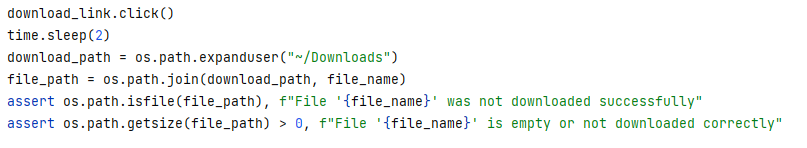
**Test report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 2** | | | | | |
| **Project Name: Zyphor** | | | | | |
| **Payment Receipt Test Case** | | | | | |
| **Test Case ID:** Test\_2 | | | **Test Designed By:** Shamjad Mazood Nazer | | |
| **Test Priority(Low/Medium/High):**  Medium | | | **Test Designed Date:** 10-05-2023 | | |
| **Module Name**: Payment | | | **Test Executed By :** Mr. Rony Tom | | |
| **Test Title :** Download receipt | | | **Test Execution Date:** 10-05-2023 | | |
| **Description:** Download the payment receipt of the student | | |  | | |
| **Pre-Condition:** User must be paid to the system for getting receipt. | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Accessing the login page |  | Login page is to be visible | Login page appeared | Pass |
| 2 | Give a real email | Email: [vishnusadas@gmail.com](mailto:vishnusadas@gmail.com) | User should be able to login | User logged in and redirected to the dashboard | Pass |
| 3 | Provide a valid password | Password: Vishnu@123 |
| 4 | Select the login button |  |
| 5 | Select payment from dashboard |  | Shows payment details | Got the payment details page. | Pass |
| 6 | Select the download button |  | File should be downloaded | File is downloaded to the system | Pass |
| **Post-Condition:** The user has successfully authenticated with the database and logged into the account. Then redirected to the payment page and download the payment receipt. | | | | | |

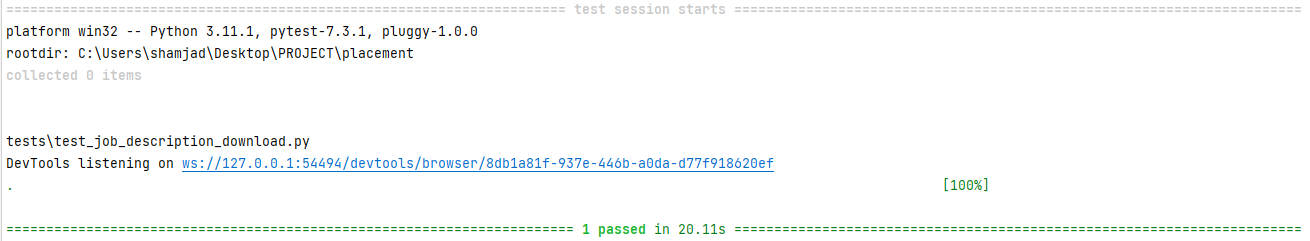
**Test Case 3:** Download Job Description File

**Code**

****

****

**Screenshot**

****

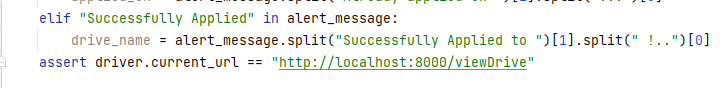
**Test report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name: Zyphor** | | | | | |
| **Download Job Description File Test Case** | | | | | |
| **Test Case ID:** Test\_3 | | | **Test Designed By:** Shamjad Mazood Nazer | | |
| **Test Priority(Low/Medium/High):**  Medium | | | **Test Designed Date:** 10-05-2023 | | |
| **Module Name**: Placement Drives | | | **Test Executed By :** Mr. Rony Tom | | |
| **Test Title :** Student Login | | | **Test Execution Date:** 10-05-2023 | | |
| **Description:** Students can download the details of the job role and company details on the given brochure. | | |  | | |
| **Pre-Condition :**User has to meet the criteria for showing the drive | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Accessing the login page |  | Login page is to be visible | Login page appeared | Pass |
| 2 | Give a real email | Email: [shamjad.nazar.20@gmail.com](mailto:shamjad.nazar.20@gmail.com) | User should be able to login | User logged in and redirected to the dashboard | Pass |
| 3 | Provide a valid password | Password: Admin@123 |
| 4 | Select the login button |  |
| 5 | Select view drives |  | Showing all the drive details | Got the drive details page. | Pass |
| 6 | Select the download button |  | File should be opened for viewing/downloading | File is opened to the used | Pass |
| **Post-Condition:** The user has successfully authenticated with the database and logged into the account. Then redirected to the view drives page and able to see/download the brochure. | | | | | |

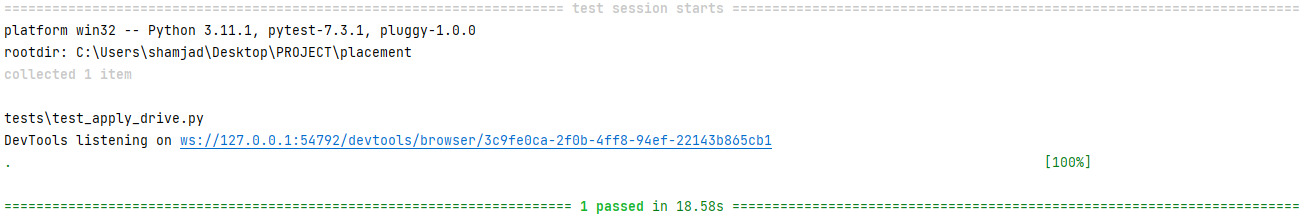
**Test Case 4:**

**Code**

****

****

**Screenshot**

****

**Test report**

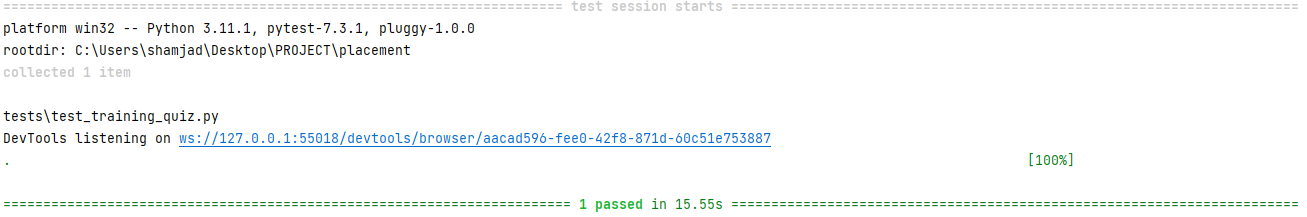
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name: Zyphor** | | | | | |
| **Applying a Drive Test Case** | | | | | |
| **Test Case ID:** Test\_4 | | | **Test Designed By:** Shamjad Mazood Nazer | | |
| **Test Priority(Low/Medium/High):**  High | | | **Test Designed Date:** 10-05-2023 | | |
| **Module Name**: Placement Drives | | | **Test Executed By :** Mr. Rony Tom | | |
| **Test Title :** Apply Job | | | **Test Execution Date:** 10-05-2023 | | |
| **Description:** When a student met the criteria for a drive, they can apply. | | |  | | |
| **Pre-Condition :**User has to meet the criteria for applying the drive | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Accessing the login page |  | Login page is to be visible | Login page appeared | Pass |
| 2 | Give a real email | Email: [shamjad.nazar.20@gmail.com](mailto:shamjad.nazar.20@gmail.com) | User should be able to login | User logged in and redirected to the dashboard | Pass |
| 3 | Provide a valid password | Password: Admin@123 |
| 4 | Select the login button |  |
| 5 | Select view drives |  | Showing all the drive details | Got the drive details page. | Pass |
| 6 | Select the apply button |  | View the drive and can apply the drive | The drive is shown and applied successfully | Pass |
| **Post-Condition:** The user has successfully authenticated with the database and logged into the account. User cannot apply to the drive if they are once applied | | | | | |

**Test Case 5:**

**Code**

****

**Screenshot**

****

**Test report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name: Zyphor** | | | | | |
| **Attending Training Quiz Test Case** | | | | | |
| **Test Case ID:** Test\_5 | | | **Test Designed By:** Shamjad Mazood Nazer | | |
| **Test Priority(Low/Medium/High):**  High | | | **Test Designed Date:** 10-05-2023 | | |
| **Module Name**: Quiz | | | **Test Executed By :** Mr. Rony Tom | | |
| **Test Title :** Attending training quiz | | | **Test Execution Date:** 10-05-2023 | | |
| **Description:** Student can take training quiz as much as they need. | | |  | | |
| **Pre-Condition:** User must to be prepared for the training quiz | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/**  **Fai l)** |
| 1 | Accessing the login page |  | Login page is to be visible | Login page appeared | Pass |
| 2 | Give a real email | Email: [vishnusadas@gmail.com](mailto:vishnusadas@gmail.com) | User should be able to login | User logged in and redirected to the dashboard | Pass |
| 3 | Provide a valid password | Password: Vishnu@123 |
| 4 | Select the login button |  |
| 5 | Select training quiz |  | Showing the training quiz | Got the training quiz page. | Pass |
| 6 | Select the options and submit button |  | Submit the test | Test is submitted successfully | Pass |
| **Post-Condition:** The user has successfully authenticated with the database and logged into the account. User can attend the training test as much as they needed. | | | | | |

# CHAPTER 6

# IMPLEMENTATION

## INTRODUCTION

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

1. The active user must be aware of the benefits of using the new system.
2. Their confidence in the software is built up.
3. Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running on the server. If the server object is not up running on the server, the actual process won’t take place.

## IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

1. The active user must be aware of the benefits of using the new system. Their confidence in the software is built up.
2. Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running on the server. If the server object is not up running on the server, the actual process won’t take place.

### User Training

The purpose of user training is to get the user ready to test and modify the system. The people who will be involved must have faith in their ability to contribute to the goal and benefits anticipated from the computer-based system. Training is more necessary as systems get more complicated. The user learns how to enter data, handle error warnings, query the database, call up routines that will generate reports, and execute other important tasks through user training.

### Training on the Application Software

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

### System Maintenance

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

### Hosting

Providing a platform or service for storing, serving, and managing websites, applications, or data on the internet is the process of hosting. Web hosting is a widespread sort of hosting that allows individuals and businesses to make their websites available on the internet. There are various kinds of web hosting, including managed hosting, cloud hosting, shared hosting, and VPS hosting. Given that it involves sharing server resources with other websites, shared hosting is the most cost-effective choice. When compared to dedicate hosting, PS hosting provides more control and flexibility over server resources. By using many servers to host websites and applications, cloud hosting provides scalability and high availability.

Through managed hosting, businesses may concentrate on their core competencies by offloading the technical aspects of hosting to a third-party supplier. Choosing the correct hosting company is vital, since it affects website performance, security, and uptime. Factors to consider when selecting a hosting provider include cost, reliability, support, scalability, and features. Overall, hosting is essential to helping people and businesses develop an online presence and reach a global audience.

**AWS Web hosting**

Amazon Web Services (AWS) offers a range of web hosting services to suit the requirements of businesses of all sizes. AWS provides a scalable and trustworthy infrastructure for hosting websites, web applications, and e-commerce platforms, among other things. Elastic Compute Cloud (EC2) by Amazon. Users have access to a number of hosting alternatives, including Amazon Lightsail, Amazon S3, AWS Elastic Beanstalk, and AWS Lambda. For delivering and scaling web applications, Elastic Beanstalk provides an easy-to-use infrastructure, but EC2 allows you total control over the virtual servers. Customers can run programs using serverless computing services like Lambda without needing to provision or manage servers. Lightsail is a simple, cost-effective option for people who only need a basic website or application, whereas Amazon S3 is an object storage service that can be used to store and retrieve files and static website content. Modern security solutions like SSL/TLS encryption, network firewalls, and distributed denial-of-service (DDoS) prevention are available through AWS web hosting services, ensuring the security and high availability of the online applications hosted on AWS.

**EC2 (Elastic cloud compute)**

Renting virtual computing resources, such as virtual machines (VMs) or instances, so that users can run their own applications is made possible via the Amazon Elastic Compute Cloud (EC2) web service, which is offered by Amazon Web Services (AWS). Users have flexibility and scalability with EC2 instances since they may be built up with different hardware configurations, operating systems, and networking configurations. EC2 instances are a cost-effective choice for companies and individuals who needs an on-demand computing power since they are simple to launch and terminate as needed and users just pay for the resources they use. In addition, EC2 works in conjunction with other AWS services to deliver a whole cloud computing platform.

**Procedure for hosting a website on AWS:**

**Step 1:** Log in to your AWS Management Console and launch the EC2 dashboard.

**Step 2:** To begin creating a new instance, choose "Launch Instance." To start constructing a new instance, click the "Launch Instance" button.

**Step 3:** Select the Amazon Machine Image (AMI) that will be the instance's building block. Typically, this will be an image that has been pre-configured with a particular software stack or operating system.

**Step 4:** To begin creating a new instance, choose "Launch Instance." To start constructing a new instance, click the "Launch Instance" button.

**Step 5:** Select the Amazon Machine Image (AMI) that will be the instance's building block. Typically, this will be an image that has been pre-configured with a particular software stack or operating system.

**Step 6:** Select the instance type that you want to use. This determines the CPU, memory, storage, and network capacity of your instance.

**Step 7:** Configure the instance's specifications, such as the number of instances you want to run, the network settings, and the security groups.

**Step 8:** Don't forget to add any extra volumes or storage units that your instance needs.

**Step 9:** Examine and configure any required advanced settings, such as IAM roles, placement groups, and user data scripts.

**Step 10:** After evaluating your instance's launch setup, which includes the instance type, storage, network configuration, and security groups, launch your instance.

**Hosted Website:**

**Hosted Link:** [**https://abc.000webhostapp.com**](https://abc.000webhostapp.com)

**Screenshot**

# CHAPTER 7

# CONCLUSION AND FUTURE SCOPE

## CONCLUSION

The application titled “College Placement Cell” developed is designed in such a way that any further enhancement can be done with ease. The system has the capability for easy integration with other systems. New modules can be added to the existing system with less effort. I put as much as my effort into developing this system based application titled “College Placement Cell” that is easily accessible, informative and helpful.

It has been designed in such a way that it is easy to modify, and can be updated efficiently and accurately. On realizing the importance of systematic documentation all the processes are implemented using a software engineering approach. Working in a live environment enables one to appreciate the intricacies involved in the System Development Life Cycle (SDLC). We have gained a lot of practical knowledge from this project, which we think, shall make us stand in a good state in the future. Once again I would like to thank everyone who was somehow or other related with the successful completion of this project.

* 1. **FUTURE SCOPE**

Analyzing the performance of students using Deep Learning concepts like ML. This could give an upper hand to placement cell and company authorities to analyze student’s academic strength and skills. It’s better for the students as well as the college faculty to mass inform the students about the pool recruitment and company specifications for a job offer. This proposed system would handle such a system by sending alert to all students in together via mail/drive using mail gateways. As this software is majorly used by students, it would be beneficial for them if the software is provided in a format of mobile application for quick access of account. As the current situations aware us about the recession, it is crucial for the Placement Cell to accommodate/provide placement to maximum students possible. So, the implementation of drastic measure of introducing one job for one student is required. Even though this could limit the opportunity for students it is a currently possible solution for the current situation. For calculating such a situation of dynamically finding the placed students and the otherwise the method of One student – One job policy would be implemented for better performance of placement drive.

.

# CHAPTER 8

# BIBLIOGRAPHY

### REFERENCES:

* + - Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009.
    - Roger S Pressman, “Software Engineering”, 1994.
    - CSS Cookbook by Christopher Schmitt.
    - Pankaj Jalote, “Software engineering: a precise approach”, 2006.

### WEBSITES:

* + - <https://docs.djangoproject.com/en/4.1>
    - [www.w3schools.com](http://www.w3schools.com)
    - [www.jquery.com](http://www.jquery.com)
    - <http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf>
    - [www.agilemodeling.com/artifacts/useCaseDiagram.html](http://www.agilemodeling.com/artifacts/useCaseDiagram.html.)
    - [www.agilemodeling.com/](http://www.agilemodeling.com/)
    - [www.tutorialspoint.org](http://www.tutorialspoint.org)
    - <https://getbootstrap.com/docs/4.0/getting-started/introduction/>
    - <https://stackoverflow.com/>
    - <http://www.geeksforgeeks.com/>

# CHAPTER 9

# APPENDIX

## Sample Code

views.py

from hashlib import sha256  
  
import stripe  
from django.contrib.auth.decorators import login\_required  
from django.contrib.auth.forms import SetPasswordForm  
from django.contrib.sites import requests  
from django.views import View  
from .forms import \*  
  
from django.template import loader  
  
from django.template.loader import render\_to\_string  
from django.contrib.sites.shortcuts import get\_current\_site  
from django.utils.http import urlsafe\_base64\_encode, urlsafe\_base64\_decode  
from django.utils.encoding import force\_bytes, force\_str  
from django.contrib import messages  
from django.contrib import auth  
from django.contrib.auth import authenticate, get\_user\_model  
from django.core.mail import EmailMultiAlternatives  
from datetime import \*  
  
from django.conf import settings  
from django.urls import reverse  
  
from django.shortcuts import render, redirect, get\_object\_or\_404  
from django.http import HttpResponse  
from .models import \*  
from .forms import RegisterForm, LoginForm  
from .decorators import user\_login\_required  
import random  
from placement.settings import EMAIL\_HOST\_USER  
  
from notifications.signals import notify  
  
from reportlab.lib.pagesizes import A4  
from reportlab.lib.units import inch  
from reportlab.lib import colors  
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Table, TableStyle  
from reportlab.lib.styles import getSampleStyleSheet  
from django.template.loader import get\_template  
from django.template import Context  
  
from .aiken import Aiken  
  
import pandas as pd  
from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
  
*# user model*User = get\_user\_model()  
  
  
*# Create your views here.*def ajax\_generate\_code(request):  
 print(request.GET)  
 for x in request.GET:  
 if x != '\_':  
 email = x  
 *# Generate Code and save it in a session* request.session['code'] = random.randint(111111, 999999)  
 *# Send email Functionality* text\_content = "Your Email Verification Code for Placement-Cell Registration is " + str(  
 request.session['code'])  
 msg = EmailMultiAlternatives('Verify Email', text\_content, EMAIL\_HOST\_USER, [email])  
 msg.send()  
 return HttpResponse("success")  
  
  
def register(request):  
 form = RegisterForm()  
 success = None  
 if request.method == 'POST':  
 if StudentReg.objects.filter(admino=request.POST['admino']).exists():  
 error = "This Admission number is already taken"  
 return render(request, 'campus/register.html', {'form': form, 'error': error})  
  
 if StudentReg.objects.filter(email=request.POST['email']).exists():  
 error = "This email is already taken"  
 return render(request, 'campus/register.html', {'form': form, 'error': error})  
  
 if (not 'code' in request.POST) or (not 'code' in request.session) or (  
 not request.POST['code'] == str(request.session['code'])):  
 error = "Invalid Verification Code"  
 return render(request, 'campus/register.html', {'form': form, 'error': error})  
 form = RegisterForm(request.POST)  
 new\_user = form.save(commit=False)  
 new\_user.save()  
 success = "New User Created Successfully, You may go back and try to login !"  
 return render(request, 'campus/register.html', {'form': form, 'success': success})  
  
  
def login\_view(request):  
 form = LoginForm()  
 if request.method == 'POST':  
 email = request.POST['email']  
 password = request.POST['password']  
 if StudentReg.objects.filter(email=email, password=password).exists():  
 user = StudentReg.objects.filter(email=email)  
 for std in user:  
 email = std.email  
 id = std.admino  
 print(email, id)  
 request.session['email'] = std.email  
 """This is a session variable and will remain existing as long as you don't delete this manually or   
 clear your browser cache """  
 request.session['admino'] = id  
 return redirect('campus:student')  
 error = "Credentials not match! Try again"  
 return render(request, 'campus/login.html', {'form': form, 'error': error})  
 else:  
 return render(request, 'campus/login.html', {'form': form})  
  
  
def update\_password(request):  
 email = request.session['email']  
 user = StudentReg.objects.get(email=email)  
 context = {  
 'user': user,  
 }  
 return render(request, 'campus/update\_password.html', context)  
  
  
def password\_changed(request):  
 email = request.session['email']  
 quiz\_result = QuizResult.objects.filter(email=email)  
 aiken\_result = Aiken\_Result.objects.filter(email=email)  
 print(email)  
 old\_password = StudentReg.objects.filter(email=email).values('password').get()['password']  
 print(old\_password)  
 current\_password = request.POST['current-password']  
 password = request.POST['new-password']  
 print(current\_password)  
 print(password)  
 if request.method == 'POST':  
 if current\_password == old\_password:  
 r = StudentReg.objects.get(email=email)  
 r.password = password  
 r.save()  
 print(r)  
 user = StudentReg.objects.get(email=email)  
 print('Admino: ', user.admino)  
 message = "Your password is now updated successfully!"  
 context = {  
 'quiz\_result': quiz\_result,  
 'aiken\_result': aiken\_result,  
 'message': message,  
 'user': user,  
 }  
 return render(request, 'campus/update\_password\_success.html', context)  
 else:  
 return HttpResponse(  
 "<script>alert('Current password is incorrect! Try again');window.location='/';</script>")  
 else:  
 print('not post')  
 messages.error(request, 'Please correct the error below.')  
 return render(request, 'campus/studentDashboard.html', {})  
  
  
def change\_profile\_picture(request):  
 return render(request, 'campus/student\_profile.html')  
  
  
def tpoLogin(request):  
 form = LoginForm()  
 if request.method == 'POST':  
 email = request.POST['email']  
 password = request.POST['password']  
 if Tpo.objects.filter(tpoMail=email, tpoPassword=password).exists():  
 user = Tpo.objects.get(tpoMail=email)  
 request.session['tpoMail'] = user.tpoMail  
 return redirect('adminDash')  
 return render(request, 'campus/adminLogin.html', {'form': form})  
 else:  
 return render(request, 'campus/adminLogin.html', {'form': form})  
  
  
def get\_admin(request):  
 return Tpo.objects.get(tpoMail=request.session['tpoMail'])  
  
  
@login\_required(login\_url='tpoLogin')  
def adminDash(request):  
 user = get\_admin(request)  
 return render(request, 'campus/adminDashboard.html', {'user': user})  
  
  
def tpoLogout(request):  
 if 'email' in request.session:  
 del request.session['email']   
 return redirect('campus:tpo')  
  
  
def get\_user(request):  
 return StudentReg.objects.get(email=request.session['email'])  
  
  
def home(request):  
 images = Company\_Image.objects.all()  
 if 'email' in request.session:  
 email = request.session['email']  
 user = get\_user(request)  
 print('ID: ', user.id)  
 quiz\_result = QuizResult.objects.filter(email=email)  
 aiken\_result = Aiken\_Result.objects.filter(email=email)  
 context = {  
 'user': user,  
 'quiz\_result': quiz\_result,  
 'aiken\_result': aiken\_result,  
 }  
 return render(request, 'campus/studentDashboard.html', context)  
 else:  
 return render(request, 'campus/index.html', {'images': images})  
  
  
@user\_login\_required  
def studentDash(request):  
 email = request.session['email']  
 user = get\_user(request)  
 myData = StudentReg.objects.filter(admino=user.admino)  
 quiz\_result = QuizResult.objects.filter(email=email)  
 aiken\_result = Aiken\_Result.objects.filter(email=email)  
 total\_score = 0  
 total\_questions = 0  
 for result in aiken\_result:  
 total\_score = total\_score + int(result.score)  
 total\_questions = total\_questions + int(result.total)  
 if total\_questions != 0:  
 average = (total\_score / total\_questions) \* 100  
 average\_score = round(average, 2)  
 print('correct: ', total\_score, '\nquestion: ', total\_questions, '\naverage: ', average\_score)  
 else:  
 average\_score = 0  
 print('No questions answered yet.')  
  
 context = {  
 'user': user,  
 'myData': myData,  
 'quiz\_result': quiz\_result,  
 'aiken\_result': aiken\_result,  
 'total\_score': total\_score,  
 'total\_questions': total\_questions,  
 'average\_score': average\_score,  
 *# 'quiz\_results': quiz\_results,* }  
 return render(request, 'campus/studentDashboard.html', context)  
  
  
def logout(request):  
 if 'email' in request.session:  
 request.session.clear() *# delete user session* return redirect('/')  
  
  
def tpo(request):  
 return render(request, 'campus/adminLogin.html')  
  
  
def student\_profile(request):  
 email = request.session['email']  
 user = get\_user(request)  
 myData = StudentReg.objects.get(email=user.email)  
 id = myData.id  
 details = MCAStudentDetails.objects.get(user=id)  
  
 context = {  
 'user': myData,  
 'details': details,  
 }  
 return render(request, 'campus/student\_profile.html', context)  
  
  
*# @login\_required(login\_url='login')*def updateStudentDetails(request):  
 email = request.session['email']  
 if Payment.objects.filter(email=email).exists():  
 form = MCAStudentDetails()  
 *# is\_private = request.POST.get('is\_private', False)* success = None  
 if request.method == 'POST':  
 form = MCAStudentDetails(request.POST)  
 if form.is\_valid():  
 data = form.save(commit=False)  
 data.admino = email  
 data.save()  
 success = "Updated Successfully !"  
 return render(request, 'campus/student\_details.html', {'form': form, 'success': success})  
 else:  
 return render(request, 'campus/payments.html', )  
  
  
def update\_profile(request):  
 email = request.session['email']  
 user = get\_user(request)  
 print(user)  
 myData = StudentReg.objects.filter(admino=user)  
 quiz\_result = QuizResult.objects.filter(email=email)  
 aiken\_result = Aiken\_Result.objects.filter(email=email)  
 phone1 = request.POST['phone1']  
 phone2 = request.POST['phone2']  
 address = request.POST['address']  
 pincode = request.POST['pincode']  
 district = request.POST['district']  
 print(phone1, phone2, address, pincode, district)  
 *# id = myData  
 # print(id)* if request.method == 'POST':  
 r = MCAStudentDetails.objects.get(user=user)  
 print(r.mobileNoIndian)  
 r.mobileNoIndian = phone1  
 print(r.mobileNoIndian)  
 r.alternativeNo = phone2  
 r.fullAddress = address  
 r.pincode = pincode  
 print(r.district)  
 r.district = district  
 print(r.district)  
 r.save()  
 print(r)  
 context = {  
 'user': user,  
 'myData': myData,  
 'quiz\_result': quiz\_result,  
 'aiken\_result': aiken\_result,  
 }  
 return HttpResponse(  
 "<script>alert('Details updated!'); window.location='studentDash'; </script>"  
 )  
 return render(request, 'campus/student\_profile.html')  
  
  
@login\_required(login\_url='login')  
def password\_change(request):  
 user = request.user  
 form = SetPasswordForm(user)  
 return render(request, 'password\_reset\_form.html', {'form': form})  
  
  
@user\_login\_required  
def viewDrive(request):  
 email = request.session['email']  
 if Payment.objects.filter(email=email).exists():  
 display\_drives = []  
 end\_drives = []  
 user = get\_user(request)  
 viewDrive = Drives.objects.all().order\_by('-last\_date')  
 myData = StudentReg.objects.get(email=user.email)  
 print(myData.id, myData.first\_name)  
 id = myData.id  
 try:  
 stddetails = MCAStudentDetails.objects.get(user=id)  
 except MCAStudentDetails.DoesNotExist:  
 stddetails = None  
 print(stddetails)  
 if stddetails is None:  
 return HttpResponse(  
 "<script>alert('Your details were not uploaded! Please do that first.'); "  
 "window.location='updateStudentDetails'; </script>"  
 )  
 else:  
 print('DOB: ', stddetails.DoB)  
 print('Foreign Obj: ', stddetails)  
 print('Student CGPA: ', stddetails.ugCgpa)  
 for drive in viewDrive:  
 print('Drive ID: ', drive.id)  
 print(stddetails.ugCgpa, '>=', drive.cgpa)  
 print(stddetails.ugPer, '>=', drive.ug\_percentage)  
 print(stddetails.mcaPer, '>=', drive.pg\_percentage)  
 print(stddetails.activeArrears, '>=', drive.backlog)  
 if float(stddetails.ugCgpa) >= drive.cgpa and float(stddetails.ugPer) >= drive.ug\_percentage and float(  
 stddetails.mcaPer) >= drive.pg\_percentage and int(stddetails.activeArrears) <= drive.backlog:  
 if datetime.today().date() <= drive.last\_date:  
 display\_drives.append(drive.id)  
 print('Drives Count: ', display\_drives)  
 else:  
 end\_drives.append(drive.id)  
 print('End Drives Count: ', end\_drives)  
 else:  
 print('Total Count: ', display\_drives + end\_drives)  
 if display\_drives is None:  
 error = "Sorry, Your profile was not met the Academic profile that recruiters needs. Kindly " \  
 "please wait for your turn"  
 context = {  
 'user': myData,  
 'error': error  
 }  
 return render(request, 'campus/viewDrive.html', context)  
 else:  
 context = {  
 'user': myData,  
 'viewDrive': viewDrive,  
 'display\_drives': display\_drives,  
 'end\_drives': end\_drives,  
 }  
 return render(request, 'campus/viewDrive.html', context)  
 else:  
 return render(request, 'campus/payments.html', )  
  
  
  
@user\_login\_required  
def register\_drive(request, id):  
 email = request.session['email']  
 user = get\_user(request)  
 drives = Drives.objects.all()  
 last\_date = Drives.objects.filter(id=id).values('last\_date').get()['last\_date']  
 drive\_id = Drives.objects.get(id=id)  
 *# applied\_date = ApplyDrive.objects.filter(drive\_name=id).values('applied\_on').get()['applied\_on']* myData = StudentReg.objects.get(email=user.email)  
 print('Drives: ', drives, '\nlast\_date: ', last\_date, '\ndrive\_id: ', drive\_id, '\nmyData: ', myData)  
 print('Today: ', datetime.today().date(), '\n', )  
 if datetime.today().date() <= last\_date:  
 print('Test on IF')  
 if ApplyDrive.objects.filter(drive\_name=drive\_id, user=myData).exists():  
 print('working on IF')  
 applied\_on = ApplyDrive.objects.get(drive\_name=drive\_id, user=myData).applied\_on  
 applied\_on\_str = applied\_on.strftime('%d-%m-%Y')  
 return HttpResponse(  
 "<script>alert('Already applied on {}!..'); window.location='/viewDrive'; </script>".format(  
 applied\_on\_str)  
 )  
 else:  
 print('testing on else')  
 email = StudentReg.objects.filter(email=email).values('email').get()['email']  
 print('Email: ', email)  
 r = ApplyDrive(drive\_name=drive\_id, user=myData, status=True)  
 print('drive\_name: ', drive\_id, 'user: ', myData.id, 'status: ', True)  
 r.save()  
  
 drive\_name = Drives.objects.filter(id=id).values('company\_name').get()['company\_name']  
 text\_content = "This mail is to inform that you have successfully registered for the drive " + drive\_name + "."  
 msg = EmailMultiAlternatives('Registration for the ' + drive\_name + ' Drive', text\_content, EMAIL\_HOST\_USER, [email])  
 msg.send()  
  
 context = {  
 'user': myData,  
 'drives': drives,  
 }  
 return HttpResponse("<script>alert('Successfully Applied to {} !..'); window.location='/viewDrive'; "  
 "</script>".format(drive\_id)  
 )  
 else:  
 print('testing outer else')  
 return HttpResponse("<script>alert('Currently we are not accepting any request for this Drive from {}!..'); "  
 "window.location='/viewDrive';</script>".format(last\_date.strftime('%d-%m-%Y'))  
 )  
  
  
@user\_login\_required  
def payment(request):  
 email = request.session['email']  
 user = StudentReg.objects.get(email=email)  
 if Payment.objects.filter(email=email).exists():  
 info = Payment.objects.filter(email=email).values('payment\_on').get()['payment\_on']  
 transaction\_id = Payment.objects.filter(email=email).values('transaction\_id').get()['transaction\_id']  
 context = {  
 'info': info,  
 'user': user,  
 'transaction\_id': transaction\_id,  
 }  
 print(info)  
 return render(request, 'campus/payment\_done.html', context)  
 else:  
 stripe.api\_key = settings.STRIPE\_PRIVATE\_KEY  
 session = stripe.checkout.Session.create(  
 payment\_method\_types=['card'],  
 line\_items=[{  
 'price': 'price\_1MfyQ0SIBKZt2GrFUxYh2TYT',  
 'quantity': 1,  
 }],  
 mode='payment',  
 success\_url=request.build\_absolute\_uri(reverse('campus:thanks')).replace('%20', '') + '?session\_id={CHECKOUT\_SESSION\_ID}',  
 cancel\_url=request.build\_absolute\_uri(reverse('campus:home')),  
 )  
 context = {  
 'session\_id': session.id,  
 'stripe\_public\_key': settings.STRIPE\_PUBLIC\_KEY  
 }  
 print(context)  
 return render(request, 'campus/payments.html', context)  
  
  
*# @user\_login\_required*def thanks(request):  
 session\_id = request.GET.get('session\_id')   
 print(session\_id)   
  
 *# Retrieve the session details from Stripe using the session ID* stripe.api\_key = settings.STRIPE\_PRIVATE\_KEY  
 try:  
 session = stripe.checkout.Session.retrieve(session\_id)  
 print(session)   
  
 if session.payment\_intent is not None:  
 transaction\_id = session.payment\_intent  
 email = request.session['email']  
 status = 'paid'  
 if not Payment.objects.filter(email=email).exists():  
 r = Payment(email=email, status=status, transaction\_id=transaction\_id)  
 r.save()  
 print(r)  
 else:  
 return HttpResponse("<script>alert('You just Paid your fee!');window.location='/payment';</script>")  
  
 return render(request, 'campus/thanks.html')  
 else:  
 return redirect('campus:pay\_error')  
  
 except stripe.error.InvalidRequestError as e:  
 print(e)  
 return redirect('campus:pay\_error')  
  
  
def pay\_error(request):  
 return render(request, 'campus/error.html')  
  
  
@user\_login\_required  
def generate\_receipt(request):  
 email = request.session['email']  
 first\_name = StudentReg.objects.filter(email=email).values('first\_name').get()['first\_name']  
 last\_name = StudentReg.objects.filter(email=email).values('last\_name').get()['last\_name']  
 payment\_amount = 5000  
 fetch\_date = Payment.objects.filter(email=email).values('payment\_on').get()['payment\_on']  
 transaction\_id = Payment.objects.filter(email=email).values('transaction\_id').get()['transaction\_id']  
 payment\_date = fetch\_date.date()  
 payment\_method = 'Credit card'  
 customer\_name = first\_name + ' ' + last\_name  
 response = HttpResponse(content\_type='application/pdf')  
 response['Content-Disposition'] = 'attachment; filename="payment\_receipt.pdf"'  
 doc = SimpleDocTemplate(response, pagesize=A4)  
 styles = getSampleStyleSheet()  
 elements = []  
 elements.append(Paragraph('Zyphor - Payment Receipt', styles['Heading1']))  
 elements.append(Spacer(1, 0.2 \* inch))  
 table\_data = [  
 ['Bill'],  
 ['Recipient Name:', customer\_name],  
 ['Payment Amount:', payment\_amount],  
 ['Stripe Transaction ID:', transaction\_id],  
 ['Payment Date:', payment\_date],  
 ['Payment Method:', payment\_method],  
 ]  
 table\_style = TableStyle([  
 ('BACKGROUND', (0, 0), (-1, 0), colors.grey),  
 ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),  
 ('ALIGN', (0, 0), (-1, -1), 'LEFT'),  
 ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),  
 ('FONTSIZE', (0, 0), (-1, 0), 14),  
 ('BOTTOMPADDING', (0, 0), (-1, 0), 12),  
 ('BACKGROUND', (0, 1), (-1, -1), colors.beige),  
 ('GRID', (0, 0), (-1, -1), 1, colors.black),  
 ])  
 elements.append(Table(table\_data, style=table\_style))  
 elements.append(Spacer(1, 0.5 \* inch))  
 elements.append(Paragraph('Thank You for your payment!', styles['Normal']))  
 doc.build(elements)  
  
 return response  
  
  
def quiz(request):  
 email = request.session['email']  
 if Payment.objects.filter(email=email).exists():  
 if request.method == 'POST':  
 print(request.POST)  
 questions = QuesModel.objects.all()  
 time = request.POST.get('timer')  
 score = 0  
 wrong = 0  
 correct = 0  
 total = 0  
 cgpa = 6.9  
 for q in questions:  
 total += 1  
 print(request.POST.get(q.question))  
 print('correct answer:' + q.ans)  
 print()  
 if q.ans == request.POST.get(q.question):  
 score += 1  
 correct += 1  
 else:  
 wrong += 1  
 percent = (score / total) \* 100  
 analysis = performance\_predict(correct, total, cgpa, time)  
 performance = round(analysis, 2)  
 if percent < 30:  
 performance = 'You must have to improve yourself'  
 elif percent <= 60:  
 performance = 'Above average performance. But have to improve'  
 elif 60 < percent <= 85:  
 performance = 'Awsome performance!'  
 else:  
 performance = 'Mind blowing! Congrats Champ'  
 print('performance:', performance)  
 context = {  
 'score': score,  
 'time': time,  
 'correct': correct,  
 'wrong': wrong,  
 'percent': percent,  
 'total': total,  
 'performance': performance,  
 }  
  
 r = QuizResult(email=email, score=score, time=time + ' sec', correct=correct,  
 wrong=wrong, percent=percent, total=total)  
 print('email:', r.email, '\nscore:', r.score, '\ntime:', r.time, '\ncorrect:', r.correct, '\nwrong:',  
 r.wrong, '\npercentage:', r.percent, '\ntotal:', r.total)  
 r.save()  
  
 return render(request, 'campus/result.html', context)  
 else:  
 questions = QuesModel.objects.all()  
 print(questions)  
 if not questions:  
 return HttpResponse(  
 "<script>alert('No Quiz for the practice mode. Try again later!..'); window.location='quiz\_mode'; "  
 "</script>"  
 )  
 else:  
 context = {  
 'questions': questions,  
 }  
 return render(request, 'campus/quizpage.html', context)  
 else:  
 return render(request, 'campus/payments.html', )  
  
  
@user\_login\_required  
def quiz\_mode(request):  
 email = request.session['email']  
 user = get\_user(request)  
 myData = StudentReg.objects.filter(admino=user.admino)  
 context = {  
 'user': user,  
 'myData': myData,  
 }  
 if Payment.objects.filter(email=email).exists():  
 return render(request, 'campus/quiz\_mode.html', context)  
 else:  
 return render(request, 'campus/payments.html', )  
  
  
@user\_login\_required  
def quiz\_list(request):  
 email = request.session['email']  
 quizzes = Quiz.objects.all()  
 quiz\_details = AikenFile.objects.all()  
 user = StudentReg.objects.get(email=email)  
 user1 = get\_user(request)  
 myData = StudentReg.objects.get(email=user1.email)  
  
 id = myData.id  
 context = {  
 'quizzes': quizzes,  
 'quiz\_details': quiz\_details,  
 'user': user,  
 }  
 print(quizzes)  
  
 try:  
 stddetails = MCAStudentDetails.objects.get(user=id)  
 except MCAStudentDetails.DoesNotExist:  
 stddetails = None  
 print(stddetails)  
 if stddetails is None:  
 return HttpResponse(  
 "<script>alert('Your details were not uploaded! Please do that first.'); "  
 "window.location='updateStudentDetails'; </script>"  
 )  
 else:  
 if not quizzes:  
 return HttpResponse(  
 "<script>alert('Nothing is Scheduled by the TPO!'); window.location='quiz\_mode'; </script>"  
 )  
 else:  
 return render(request, 'campus/quiz\_list.html', context)  
  
  
@user\_login\_required  
def quiz\_detail(request, id):  
 email = request.session['email']  
 quiz = Quiz.objects.get(id=id)  
 questions = quiz.question\_set.all()  
 times = AikenFile.objects.filter(id=id).values('time').get()['time']  
 start\_date = AikenFile.objects.filter(id=id).values('start\_date').get()['start\_date']  
 end\_date = AikenFile.objects.filter(id=id).values('end\_date').get()['end\_date']  
 print('start: ', start\_date, '\nend: ', end\_date)  
 if datetime.today().date() > end\_date:  
 print('if : ', datetime.today().date())  
 return HttpResponse("<script>alert('This quiz is longer available from {}!..'); window.location='/quiz\_list'; "  
 "</script>".format(  
 end\_date.strftime('%d-%m-%Y'))  
 )  
 elif datetime.today().date() < start\_date:  
 print('else : ', datetime.today().date())  
 return HttpResponse(  
 "<script>alert('This quiz will only be opened on {}!..'); window.location='/quiz\_list'; </script>".format(  
 start\_date.strftime('%d-%m-%Y'))  
 )  
 attempt\_counter = Aiken\_Result.objects.filter(  
 quiz\_id=id, email=email  
 ).values\_list("counter", flat=True).first()  
  
 context = {  
 "quiz": quiz,  
 "questions": questions,  
 "times": times,  
 }  
  
 if not attempt\_counter:  
 return render(request, "campus/quiz\_details.html", context)  
 else:  
 return HttpResponse(  
 "<script>alert('Already attended this quiz!..'); window.location='/quiz\_list'; </script>"  
 )  
  
  
@user\_login\_required  
def submit\_quiz(request, id):  
 email = request.session['email']  
 user = get\_user(request)  
 quiz = get\_object\_or\_404(Quiz, pk=id)  
 myData = StudentReg.objects.get(email=user.email)  
 sid = myData.id  
 student = MCAStudentDetails.objects.get(user=sid)  
 quiz\_name = AikenFile.objects.filter(id=id).values('id').get()['id']  
 print('quiz\_name: ', quiz\_name)  
 print('Quiz ID: ', quiz)  
 if request.method == 'POST':  
 quiz\_id = id  
 score = 0  
 quiz\_name = quiz  
 time = request.POST.get('total\_sec')  
 correct = 0  
 wrong = 0  
 percent = 0  
 total = 0  
 questions\_with\_answers = []  
  
 for question in quiz.question\_set.all():  
 answer\_id = request.POST.get(f'question\_{question.id}')  
 print('Answer ID :', answer\_id)  
 total += 1  
 if answer\_id:  
 answer = get\_object\_or\_404(Answer, pk=answer\_id)  
 print('Answer :', answer)  
 if answer.is\_correct:  
 correct += 1  
 print('Correct Answer :', answer.is\_correct)  
 score += 1  
 print(score)  
 questions\_with\_answers.append({  
 'question': question,  
 'answer': answer,  
 'is\_correct': True,  
 })  
 else:  
 wrong += 1  
 print('Wrong Answer')  
 questions\_with\_answers.append({  
 'question': question,  
 'answer': answer,  
 'is\_correct': False,  
 })  
 else:  
 questions\_with\_answers.append({  
 'question': question,  
 'answer': None,  
 'is\_correct': False,  
 })  
 percent = (score / total) \* 100  
 print('Total Mark : ', score)  
 counter = 1  
 print('counter: ', counter)  
 r = Aiken\_Result(quiz\_id=quiz\_name, email=email, score=score, quiz\_name=quiz\_name, time=time,  
 correct=correct, wrong=wrong, percent=percent, total=total, counter=counter)  
 r.save()  
 print(r)  
 pg\_per = student.mcaPer  
 pg\_cgpa = student.mcaAggregateCgpa  
 ug\_per = student.ugPer  
 ug\_cgpa = student.ugCgpa  
 hse\_per = student.hsePer  
 sslc\_per = student.sslcPer  
 print(pg\_per, pg\_cgpa, ug\_per, ug\_cgpa, hse\_per, sslc\_per, percent)  
 analysis = placement\_prediction(pg\_per, pg\_cgpa, ug\_per, ug\_cgpa, hse\_per, sslc\_per, percent)  
 context = {  
 'quiz\_id': id,  
 'quiz': quiz,  
 'email': email,  
 'score': score,  
 'quiz\_name': quiz\_name,  
 'time': time,  
 'correct': correct,  
 'wrong': wrong,  
 'percent': percent,  
 'total': total,  
 'chances': analysis,  
 'questions\_with\_answers': questions\_with\_answers,  
 }  
 return render(request, 'campus/quiz\_results.html', context)  
 return render(request, 'campus/quiz\_list.html', {'quiz': quiz})  
  
  
def sent\_message(request):  
 return render(request, 'chat\_to\_admin')  
  
  
def performance\_predict(correct, total, cgpa, time):  
 print(correct, total, cgpa, time)  
 df = pd.read\_csv('static/csv/Student.csv')  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(df.drop('output', axis=1), df['output'], test\_size=0.2,  
 random\_state=0)  
 regressor = LinearRegression()  
 regressor.fit(X\_train, y\_train)  
 y\_pred = regressor.predict(X\_test)  
 print(y\_pred)  
 from sklearn.metrics import mean\_squared\_error, r2\_score  
 print('Mean squared error: %.2f' % mean\_squared\_error(y\_test, y\_pred))  
 print('Coefficient of determination (R^2): %.2f' % r2\_score(y\_test, y\_pred))  
 print('Accuracy : %.2f' % (r2\_score(y\_test, y\_pred) \* 100))  
 print('prediction: ', y\_pred)  
  
 return r2\_score(y\_test, y\_pred) \* 100  
  
  
def chat\_to\_admin(request):  
 email = request.session['email']  
 print(email)  
 user = StudentReg.objects.get(email=email)  
 context = {  
 'user': user,  
 }  
 return render(request, 'campus/chat.html', context)  
  
  
import os  
import pickle  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
  
  
def placement\_prediction(pg\_per, pg\_cgpa, ug\_per, ug\_cgpa, hse\_per, sslc\_per, quiz\_per):  
 print('pg\_per: ', pg\_per, '\npg\_cgpa: ', pg\_cgpa, '\nug\_per: ', ug\_per, '\nug\_cgpa: ', ug\_cgpa, '\nhse\_per: ',  
 hse\_per, '\nsslc\_per: ', sslc\_per, '\nquiz\_per: ', quiz\_per)  
 pickle\_file = 'static/csv/model.pickle'  
  
 if os.path.exists(pickle\_file):  
 with open(pickle\_file, 'rb') as f:  
 model = pickle.load(f)  
 else:  
 data = pd.read\_csv("static/csv/2020-Student-DB.csv")  
  
 X = data.drop("output", axis=1)  
 y = data["output"]  
  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
 model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
  
 model.fit(X\_train, y\_train)  
  
 with open(pickle\_file, 'wb') as f:  
 pickle.dump(model, f)  
  
 new\_student\_scores = [pg\_per, pg\_cgpa, ug\_per, ug\_cgpa, hse\_per, sslc\_per, quiz\_per]  
 prediction = model.predict([new\_student\_scores])  
 print("Placement Prediction: ", prediction[0]\*100)  
  
 return prediction[0]\*100

aiken\_file\_parsing

@admin.register(AikenFile)  
class AikenFileAdmin(admin.ModelAdmin):  
 list\_display = ['id', 'name', 'uploaded\_on', 'time', 'start\_date', 'end\_date', 'file']  
 ordering = ['id']  
  
 def save\_model(self, request, obj, form, change):  
 super().save\_model(request, obj, form, change)  
 file\_path = obj.file.path  
 with open(file\_path, 'r') as f:  
 lines = f.readlines()  
  
 if len(lines) < 6:  
 return  
  
 quiz\_title = lines[0].strip()  
 if not quiz\_title:  
 return  
  
 quiz = Quiz.objects.create(title=quiz\_title)  
 print(quiz)  
  
 for i in range(1, len(lines), 6):  
 if i + 5 > len(lines):  
 return  
  
 question\_text = lines[i].strip()[:]  
 if not question\_text:  
 return  
  
 question = Question.objects.create(quiz=quiz, text=question\_text)  
 print(question)  
  
 answers = []  
 for j in range(i + 1, i + 5):  
 if j >= len(lines):  
returnanswer\_text = lines[j].strip()[3:].lstrip('. ')  
 if not answer\_text:return  
  
 print('line[j] : ', lines[j].strip()[3:])  
 print('start with : ', lines[j].startswith('\*'))  
  
 is\_correct = lines[j].startswith('\*')answer = Answer(question=question, text=answer\_text, is\_correct=is\_correct)  
 answers.append(answer)  
 print(answers)  
  
 if len(answers) == 4:  
 Answer.objects.bulk\_create(answers)  
  
 def has\_change\_permission(self, request, obj=None):  
 return False

Student\_data\_export\_CSV

def export\_to\_csv(modeladmin, request, queryset):  
 response = HttpResponse(content\_type='text/csv')  
 response['Content-Disposition'] = 'attachment; filename="students data.csv"'  
  
 writer = csv.writer(response)  
 writer.writerow(['First Name', 'Last Name', 'Email', 'Contact Number', 'SSLC Percentage', 'HSE Percentage', 'Name of UG', 'UG CGPA', 'UG Percentage', 'MCA Aggregate CGPA', 'MCA Percentage', 'Active Arrears', 'History of Arrears', 'Exams Not Attended'])  
  
 for apply\_drive in queryset:  
 student\_reg = apply\_drive.user  
 mca\_details = student\_reg.mcastudentdetails  
  
 row = [  
 student\_reg.first\_name,  
 student\_reg.last\_name,  
 student\_reg.email,  
 mca\_details.mobileNoIndian,  
 mca\_details.sslcPer,  
 mca\_details.hsePer,  
 mca\_details.nameOfUG,  
 mca\_details.ugCgpa,  
 mca\_details.ugPer,  
 mca\_details.mcaAggregateCgpa,  
 mca\_details.mcaPer,  
 mca\_details.activeArrears,  
 mca\_details.historyOfArrears,  
 mca\_details.examsNotAttended  
 ]  
  
 writer.writerow(row)  
  
 return response  
  
  
export\_to\_csv.short\_description = "Export selected drives to CSV"

## Screen Shots

## All screenshots

**Attach Plagiarism Report**