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

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

1.1 PROJECT OVERVIEW

The Zyphor (Placement Management System) Project is a web-based programme that can be used both inside and outside the organisation with the proper login credentials. This system may be used as an application by the college's TPO to manage student placement information. This system was built using HTML, CSS, Bootstrap, JavaScript, the Python Programming Language, and a SQLite database. The system is designed and developed for the automation of campus drives and the generation of reports for the training and placement department. An online application may be accessible throughout our organisation with the proper credentials.

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.

Teacher

1. Login.
2. View student

1.2 PROJECT SPECIFICATION

A college placement cell is an essential part of every college. Final-year students attend interviews with several firms invited by the placement officer. An online system where the placement officer may publish forthcoming placement drives is required for this. Students who want to participate in these placement drives must be able to register with the system for a modest fee and get notifications for any placement drives that meet their qualifications. This Python-Django web application, Zyphor Project, fully supports the college placement cell's capabilities in online, where students will receive only the drives that are based on their academic achievement. Students can take the Mock test that the TPO has uploaded, and the system will display each student's performance.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System study is a broad word that refers to a methodical, organised approach to discovering and addressing issues. We call it system Study process lifecycle methodology because it refers to four critical stages in the lifespan of any corporate information systems. The system is thoroughly researched and analysed.

The process of breaking anything down into bits in order to understand the whole. The notion of system analysis encompasses not only the process of analysis but also that of synthesis, which indicates the act of bringing together to produce a new whole. All actions related with each step of the life cycle must be completed, monitored, and documented. As a result, we define system analysis as the performance, administration, and recording of activities associated with the life cycle stages of a computer-based business system. During the study phase, the project is thoroughly researched and a comprehensive image of the project is formed. During the design phase, the input, output, and table designs are created. The development phase is where the physical design of the input-output displays and system coding take place. System implementation is the process of putting the system into action by doing the appropriate tests.

2.2 EXISTING SYSTEM

The old system has to be thoroughly reviewed and analysed in order to elicit system requirements and identify the elements, inputs, outputs, subsystems, and procedures. This is the system analysis. Records, slips, procedures, and rules, among other things, were exhaustively investigated. The existing system was researched with full participation from the personnel who now manage the system.

2.2.1 NATURAL SYSTEM STUDIED

This system has the following flaws. Because the technique comprises of several phases, direct contact is not feasible. Communication and file management are difficult tasks. The search for an existing system is not feasible. It is not possible to update often.

2.2.2 DESIGNED SYSTEM STUDIED

This system maintains track of student placement information. It improves on the present system. It has the capacity to store the student's information, which reduces the amount of human labor necessary. It will save time and energy that would otherwise be spent on report creation and data collection.

2.3 DRAWBACKS OF EXISTING SYSTEM

- A huge number of distinct registries are required.
- The possibility of losing information.
- Requires a considerable amount of storage space.
- Reports were unable to be completed on time.
- It was difficult to use and maintain paper files, registrations, and other written materials.
- The system's security was dependent on the people who worked with it.
- Make performance evaluations of individual pupils harder.

2.4 PROPOSED SYSTEM

The "Zyphor Project" web application is focused on student placement/recruitment. Students can add their academic information, extracurricular activities, and achievements until the drive's deadline. There are three types of users in this system: administrators (TPOs), teachers, and students. The administrator may review the reports as well as the student's academic performance status and filter/sort them in the system for mailing purposes.

2.5 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

3.1 FEASIBILITY STUDY

A feasibility study evaluates a system proposal's workability, influence on the organisation, ability to fulfil user demands, and effective use of resources. The goal of the feasibility study is to gain an understanding of the system's breadth.

A project's viability can be determined using technical considerations, economic factors, or both. A feasibility study is recorded with a report that details all of the project's implications. The feasibility study must be evaluated, and risk analysis is linked in many ways. If the project study is excellent, the feasibility of generating high-quality software is diminished.

The key factors considered during the feasibility study are:

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

3.1.1 Economic Feasibility

Economic analysis is the most often utilised tool for assessing a system's efficacy. The most significant examination of the project's economic basis is a cost-benefit analysis. Cost-benefit analysis delineates project development costs and compares them against concrete and intangible system advantages. The characteristics of the system to be constructed, the project's relative location, and the projected return on investment all influence this sort of study. The benefits of a new system are always calculated in relation to the old method of operation.

Economic feasibility considers the financial impact on the organization of implementing the new system. Not only are the expenses of hardware, software, and so on addressed, but also the form of cost savings. The project, once implemented, will undoubtedly be helpful since it will save manual labor and boost work speed.

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.

3.1.2 Technical Feasibility

A variety of technical difficulties are commonly highlighted during the feasibility stage of the inquiry. A study of function, performance, and limits enabled me to create an acceptable system. This system requires the following software:

1. Python
2. SQLite

3.1.3 Behavioral Feasibility

Proposed projects are only useful if they can be transformed into information systems that fulfil the organization's operational needs. This feasibility test determines if the system will work when completed and meets all operational requirements. It was the hardest work for me, but I completed it quickly. The system is deemed practical because this package is technically, economically, and operationally practicable. The proposed system is evaluated based on the information gathered, recommendations, and reasoning. As a result, the decision is made to proceed with the project.

3.1.4 Feasibility Study Questionnaire

1. Who are the users of the system?

   A) TPO, Student, Class Teachers.
2. 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.
3. 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.
4. 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.

5. What is the role of HOD?

   A) They can view the performance of their own students in the department.
6. 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.
7. What if a student got already placed?

   A) They cannot receive any further updates about any new drives.
8. 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.
9. 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.
10. Additional requirements?

    A) TPO can sort the students based on their academic performance, and send the mail to those students about the drive.
11. Are there any measures in place to assist students in preparing for job interviews and improving their soft skills?

    A) Make a training platform for the students which they can perform aptitude assessment and can analyze their performance by themselves.
12. How do you handle situations where students are not placed or do not receive satisfactory job offers?

    A) 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.
13. 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?

    A) 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.
14. Should there be any time restrictions or deadlines for completing the mock tests?

    A) Yes, like the actual online aptitude test we can do the same within our system too.

15. Would you like to include explanations or solutions for the questions after the mock test?
    A) 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 is investigating the present system to see how it operates and where improvements might be made. A detailed understanding of the present system is essential for implementing necessary modifications. Proper planning and data collecting fulfil the objective. The popularity of this document is to detail all the requirements for the popularity of the programme College Placement Cell. We cannot visit every firm or office because of our hectic schedules. This issue will administer this website. It will assist in avoiding corrections and missing data.

3.2.1 Hardware Specification

Processor	-	Dual Core
RAM	-	2 GB RAM
Hard disk	-	120 GB

3.2.2 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

3.3 SOFTWARE DESCRIPTION

3.3.1 INTRODUCTION TO PYTHON

Python is a dynamic, high-level, free, open-source, and interpreted programming language. It may be used for both object-oriented and procedural programming. Python is a dynamically typed language, thus we don't need to define variable types. For example, x=10, where x might be anything like String, int etc.

Features in Python

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

1) Easy to code:

   Python is an advanced programming language. Python is a fairly simple language to learn when compared to other languages such as C, C#, JavaScript, and Java. Python programming is fairly simple, and anyone can master the basics in a matter of hours or days. It is also a language for developers.

2) Free and Open Source:

   Python is a free language that may be downloaded from the official website. Because it is open-source, the source code is also accessible to the public. So you may download it, use it, and share it.

3) Object-Oriented Language:

   Object-Oriented programming is a core element of Python. Python supports object oriented programming and concepts such as classes, object encapsulation, and so on.

4) High-Level Language:

   Python is a high-level programming language. When we build Python programs, we don't need to know the system architecture or manage the memory.

5) Python is Portable Language

   Python is also a portable programming language. For example, if we have Python code for Windows and wish to execute it on other platforms such as Linux, UNIX, and Mac, we do not need to update it; we can run this code on any platform.

6) Large Standard Library:

   Python offers a big standard library that provides a comprehensive collection of modules and functions, eliminating the need to develop your own code for everything. Python includes several libraries, such as regular expressions, unit testing, and web browsers.

3.3.2 SQLite

This SQLite tutorial will teach you all you need to know to get started with SQLite. In this lesson, you will master SQLite step by step with considerable hands-on practice.

This SQLite course is intended for developers who wish to use SQLite as a back-end database or to handle structured data in desktop, web, and mobile applications.

SQLite is a zero-configuration, self-contained, stand-alone transaction relational database engine meant to be integrated in an application.

CHAPTER 4
SYSTEM DESIGN

4.1 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.

4.2 UML DIAGRAM

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

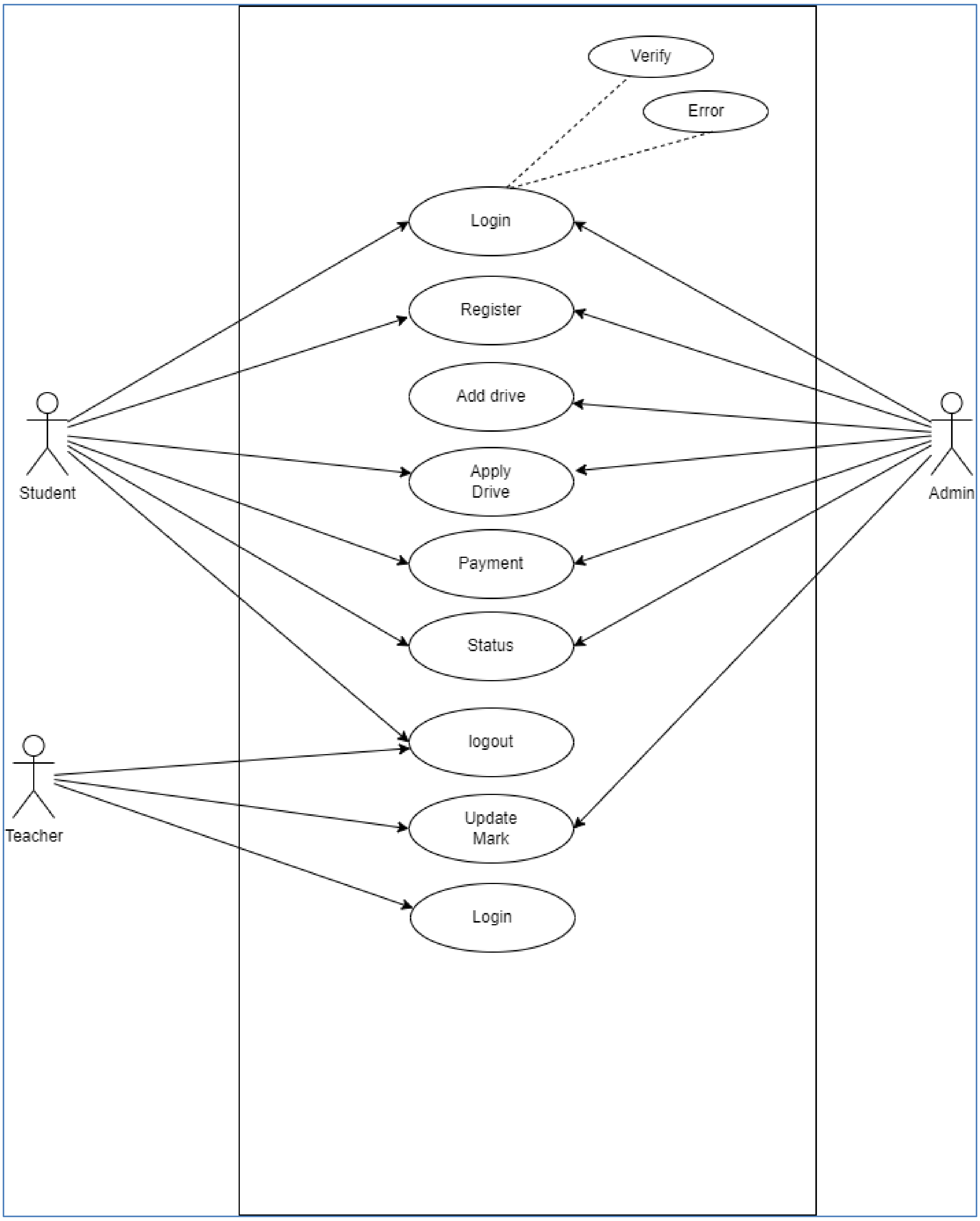
4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between system components. An approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" 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.


Verify
Error
Login
Register
Add drive
Apply Drive
Payment
Status
logout
Update Mark
Login
Student
Teacher
Admin

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 –

i. **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.

ii. **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.

iii. **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.

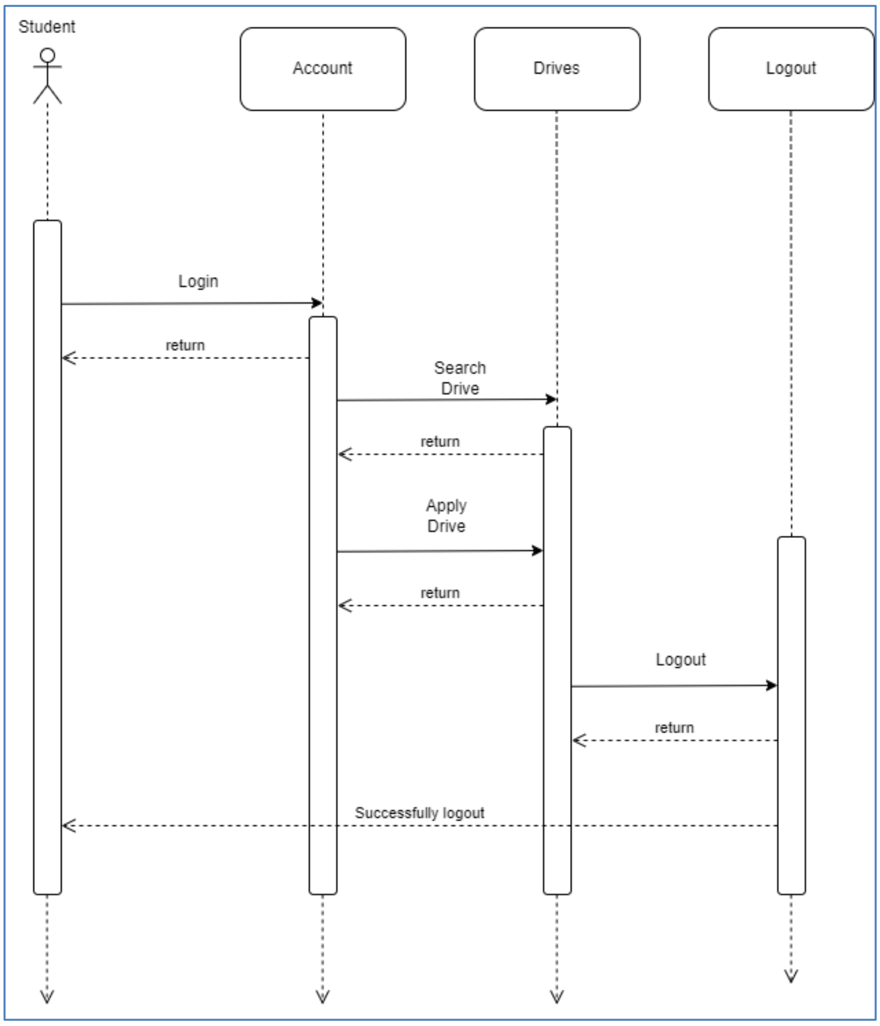
Messages can be broadly classified into the following categories:

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

iv. **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.

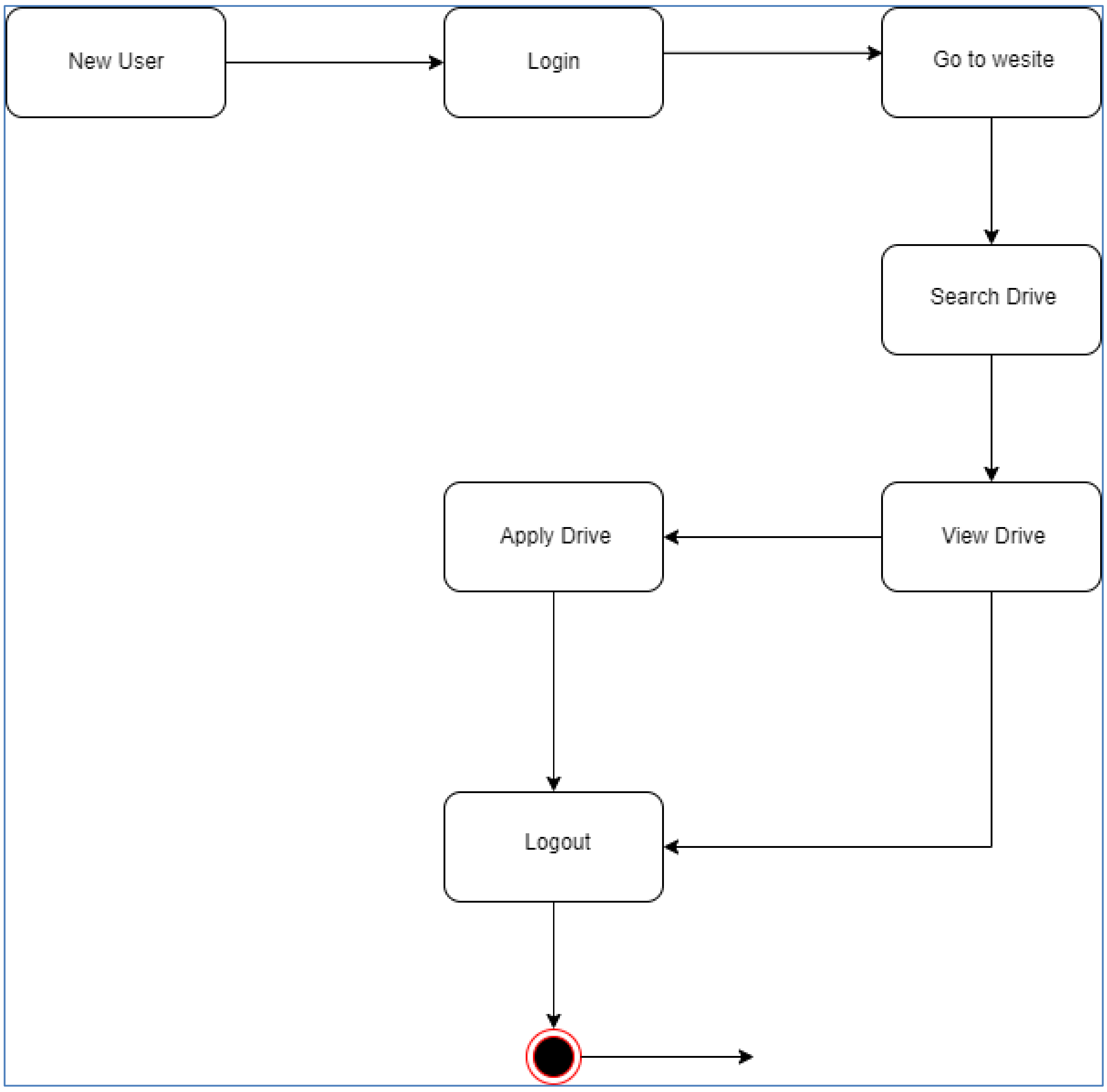
Uses of sequence diagrams –

- 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.



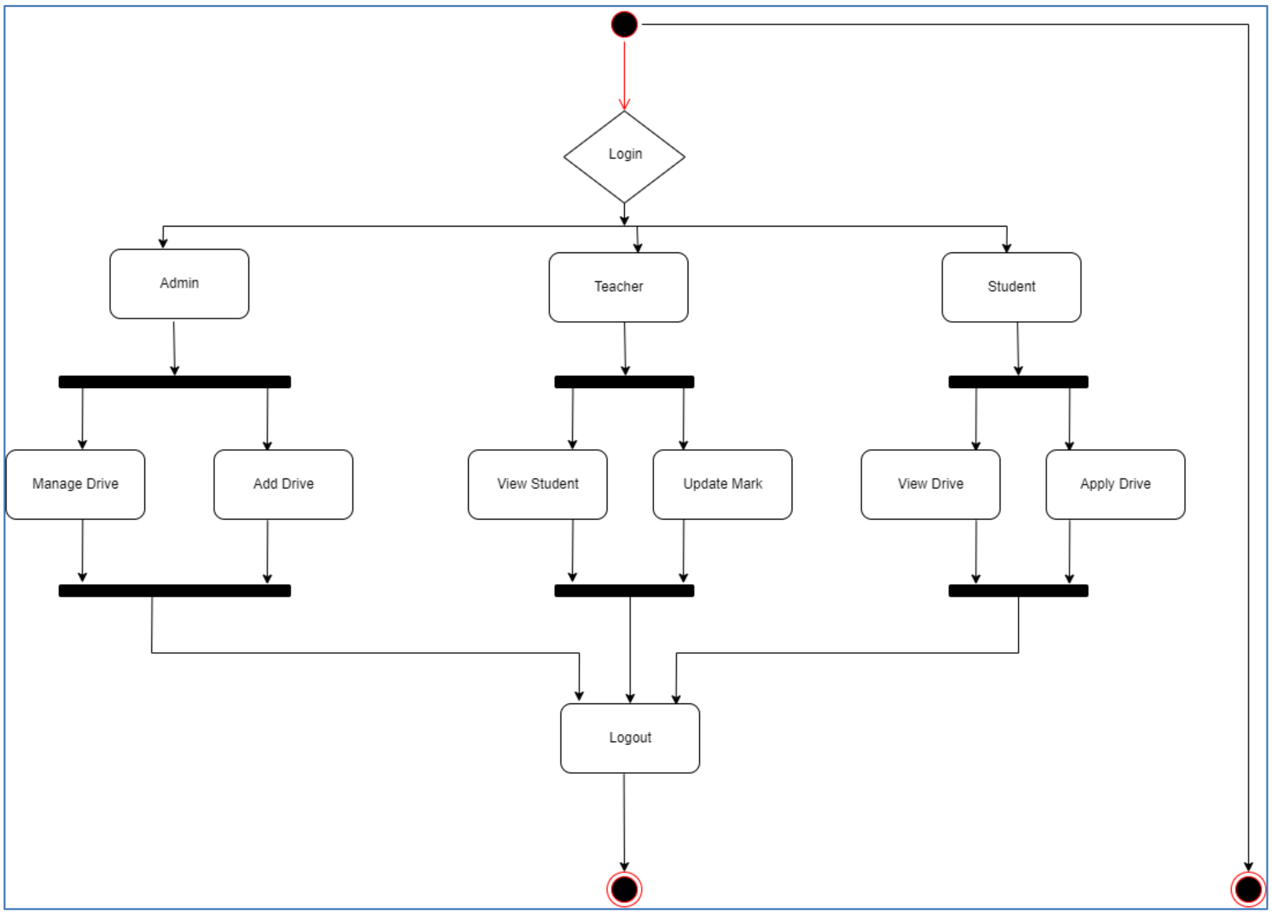
4.2.3 State Chart Diagram

A state chart diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML)



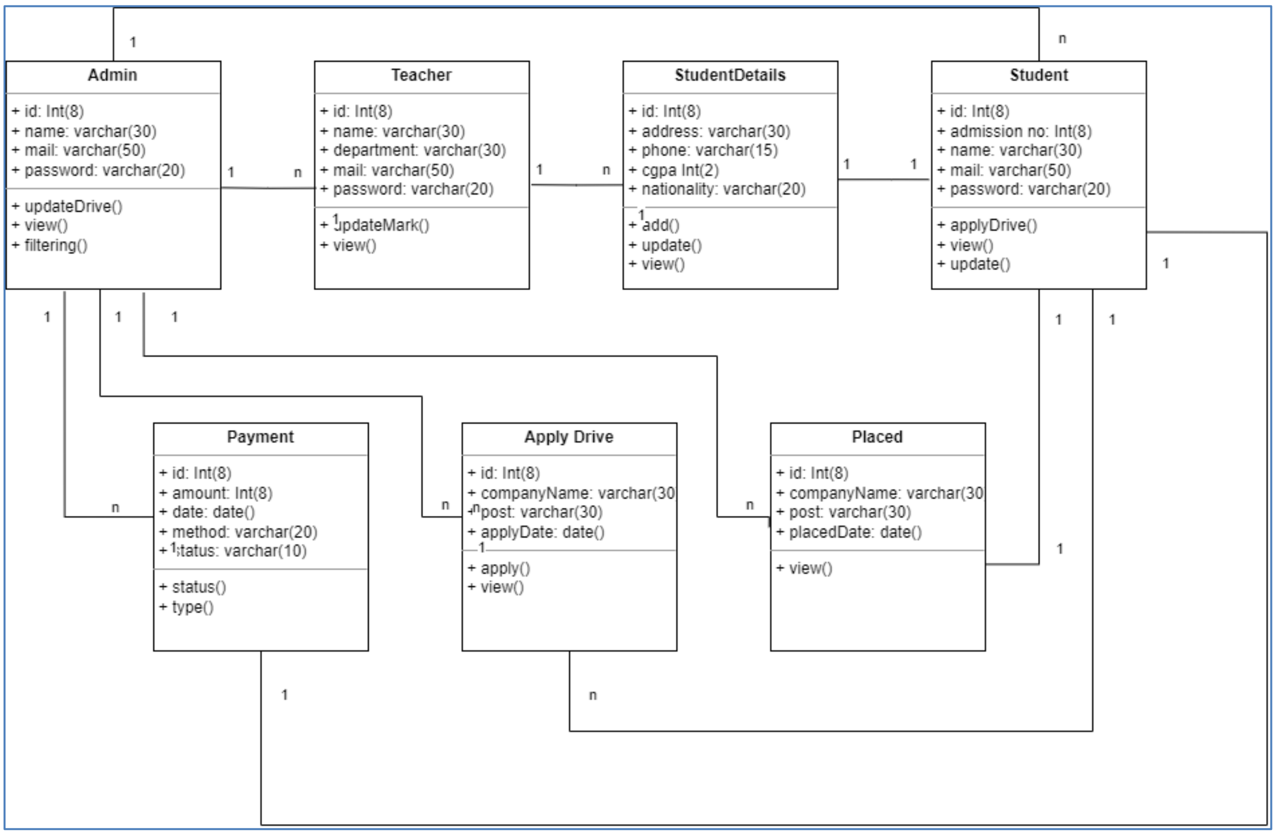
4.2.4 Activity Diagram

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.



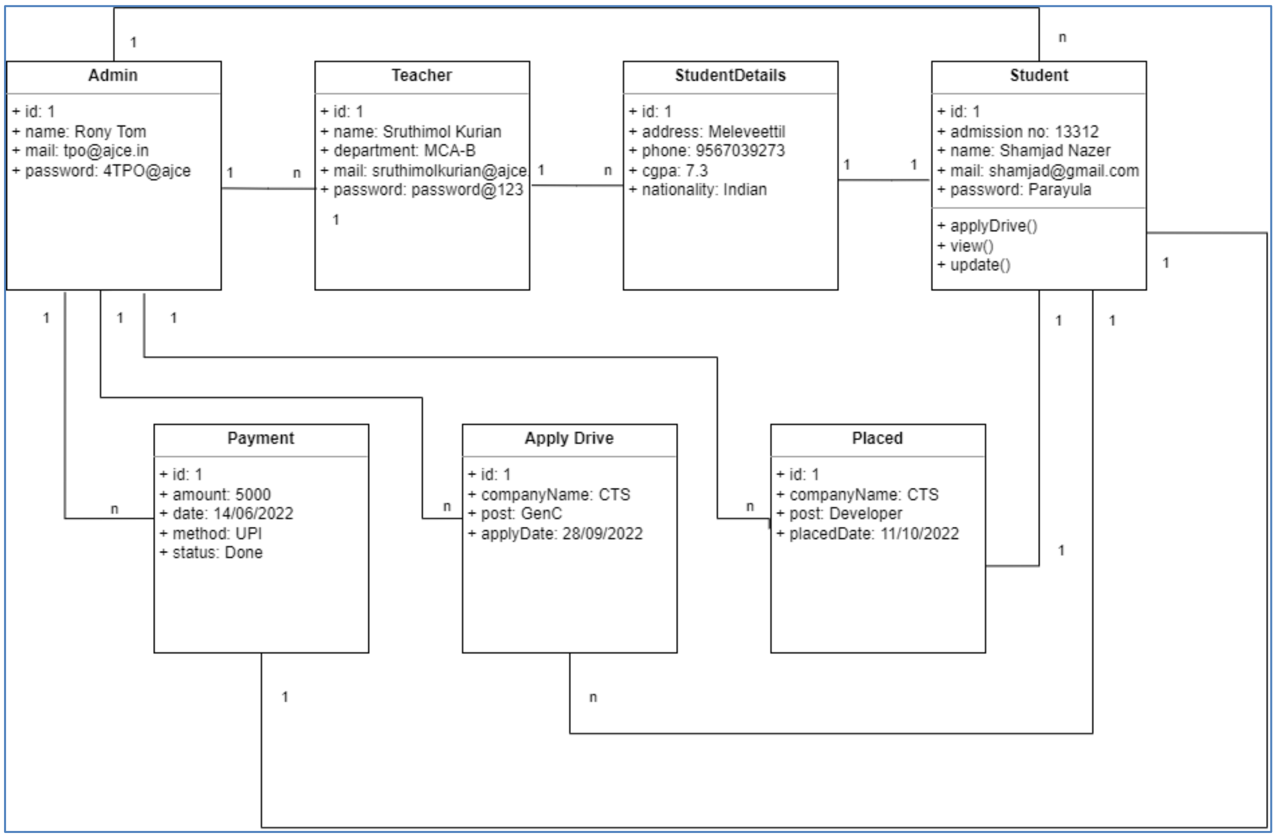
4.2.5 Class Diagram

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.



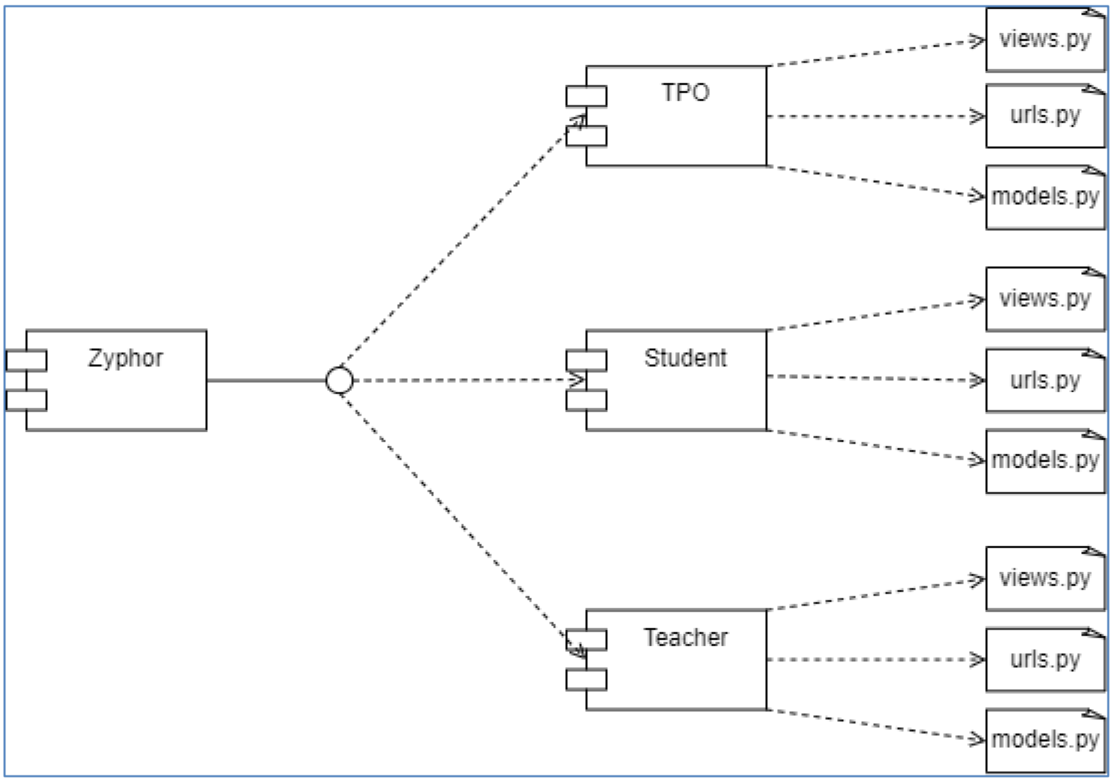
4.2.6 Object Diagram

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.



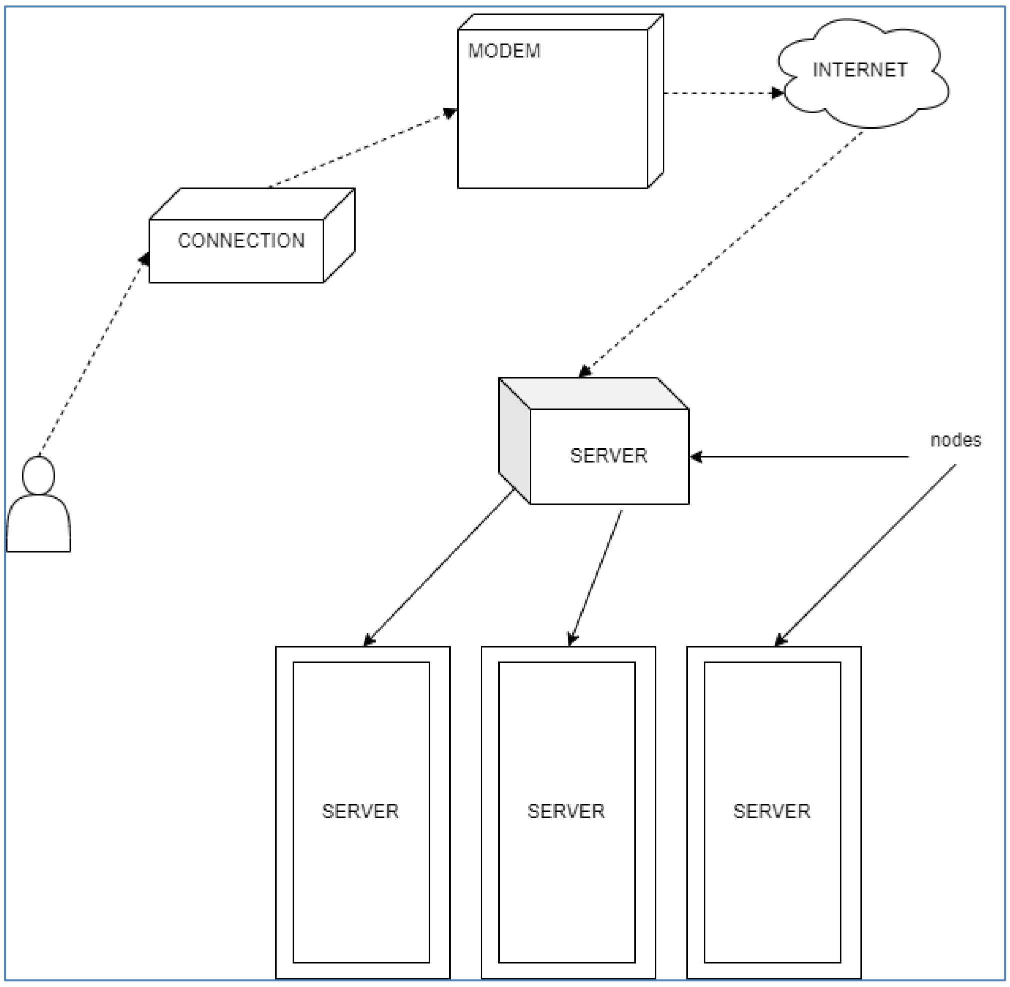
4.2.7 Component Diagram

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



4.2.8 Deployment Diagram

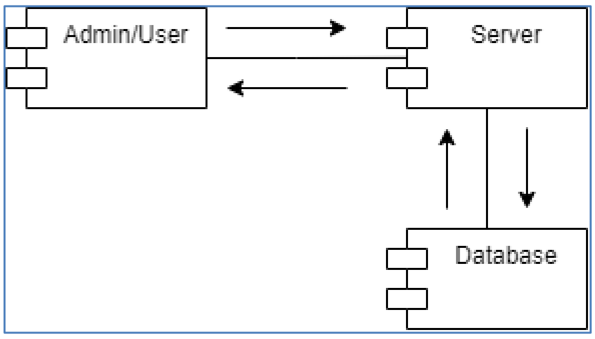
An execution architecture of a system, containing nodes like hardware or software execution environments, and the middleware linking them, is shown in a deployment diagram, a form of UML diagram. Typically, deployment diagrams are used to represent the actual hardware and software of a system. By using 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

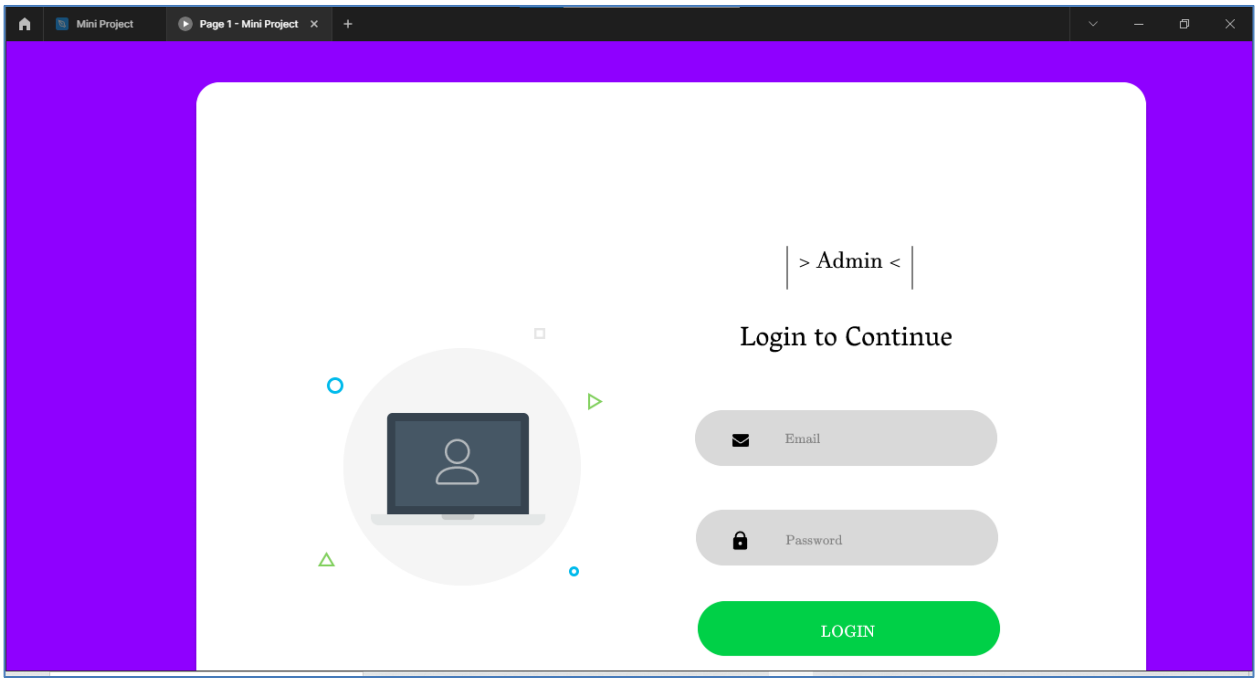
A collaboration diagram, also known as a communication diagram, is a form of UML (Unified Modelling Language) diagram that depicts the interactions and relationships between objects or actors in a system. It emphasizes the dynamic component of the system by demonstrating how items collaborate to attain a certain feature or complete a specific activity.

The items or players participating in the interaction are represented by lifelines (vertical lines) in a collaboration diagram. Messages transmitted between these lifelines are depicted as arrows with labels reflecting the message's substance. The order of messages is shown by the order in which they appear on the diagram.

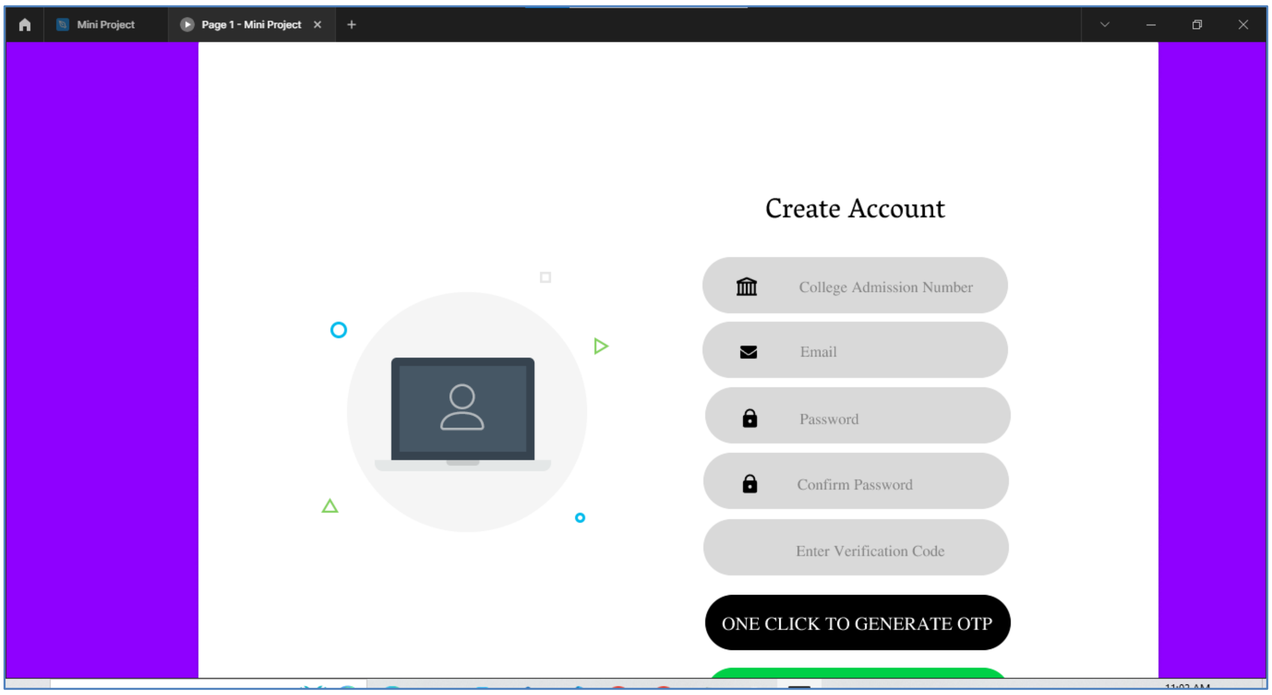


4.3 USER INTERFACE DESIGN USING FIGMA

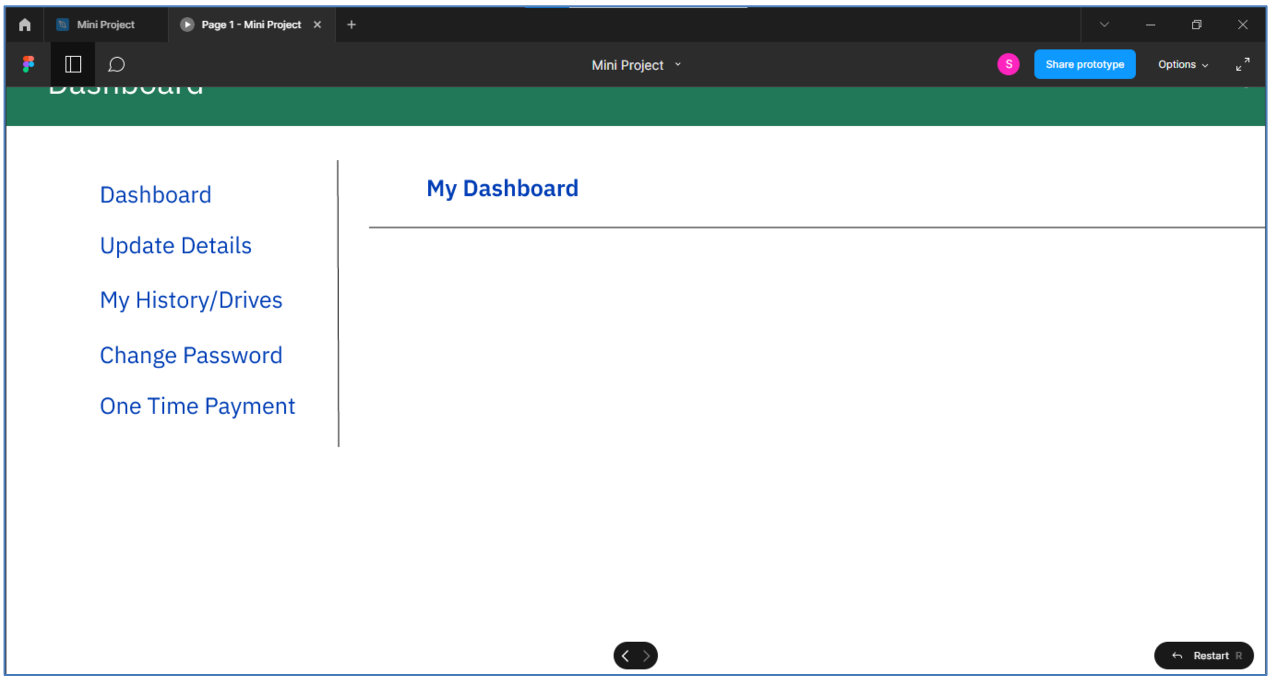
Form Name: Login



Form Name: Registration Form



Form Name: Student Dashboard



Dashboard

My Dashboard

Dashboard

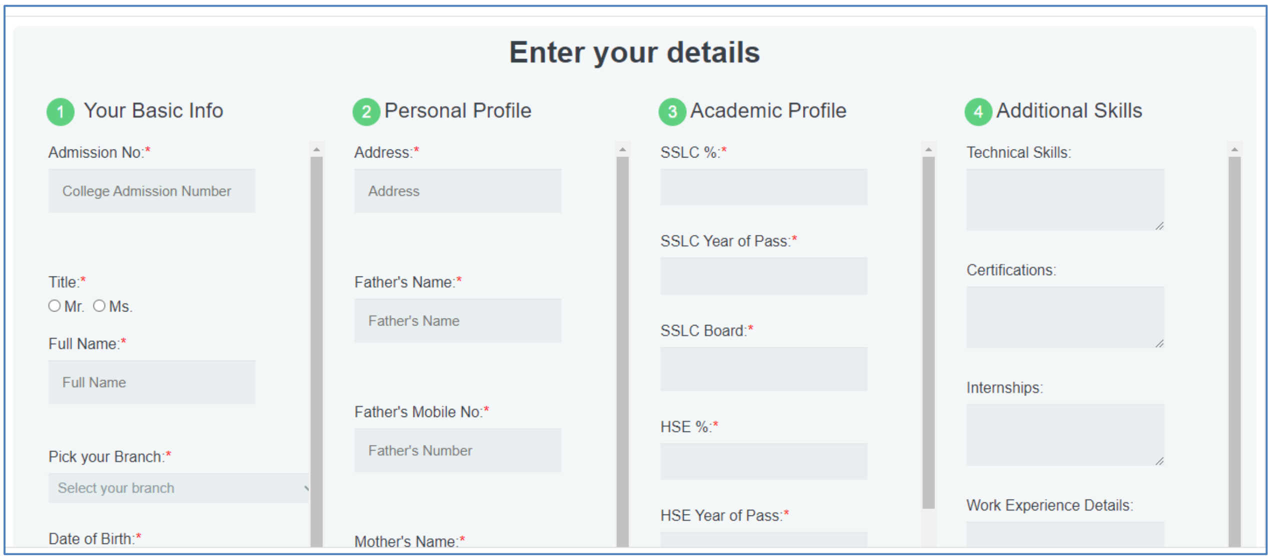
Update Details

My History/Drives

Change Password

One Time Payment

Form Name: Student Details



Enter your details

1 Your Basic Info

Admission No:*

College Admission Number

Title:*

○ Mr. ○ Ms.

Full Name:*

Full Name

Pick your Branch:*

Select your branch

Date of Birth:*

2 Personal Profile

Address:*

Address

Father's Name:*

Father's Name

Father's Mobile No:*

Father's Number

Mother's Name:*

3 Academic Profile

SSLC %:*

SSLC Year of Pass:*

SSLC Board:*

HSE %:*

HSE Year of Pass:*

4 Additional Skills

Technical Skills:

Certifications:

Internships:

Work Experience Details:

4.3 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.

4.4 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

2. **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

3. **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

4. **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

5. **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

6. **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

7. 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

8. 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

9. 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

10. 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

11. 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

12. 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

5.1 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.

5.2 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

5.2.1 Unit Testing

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as 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.

5.2.2 Integration Testing

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

5.2.3 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.

5.2.4 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.

5.2.5 Automation Testing

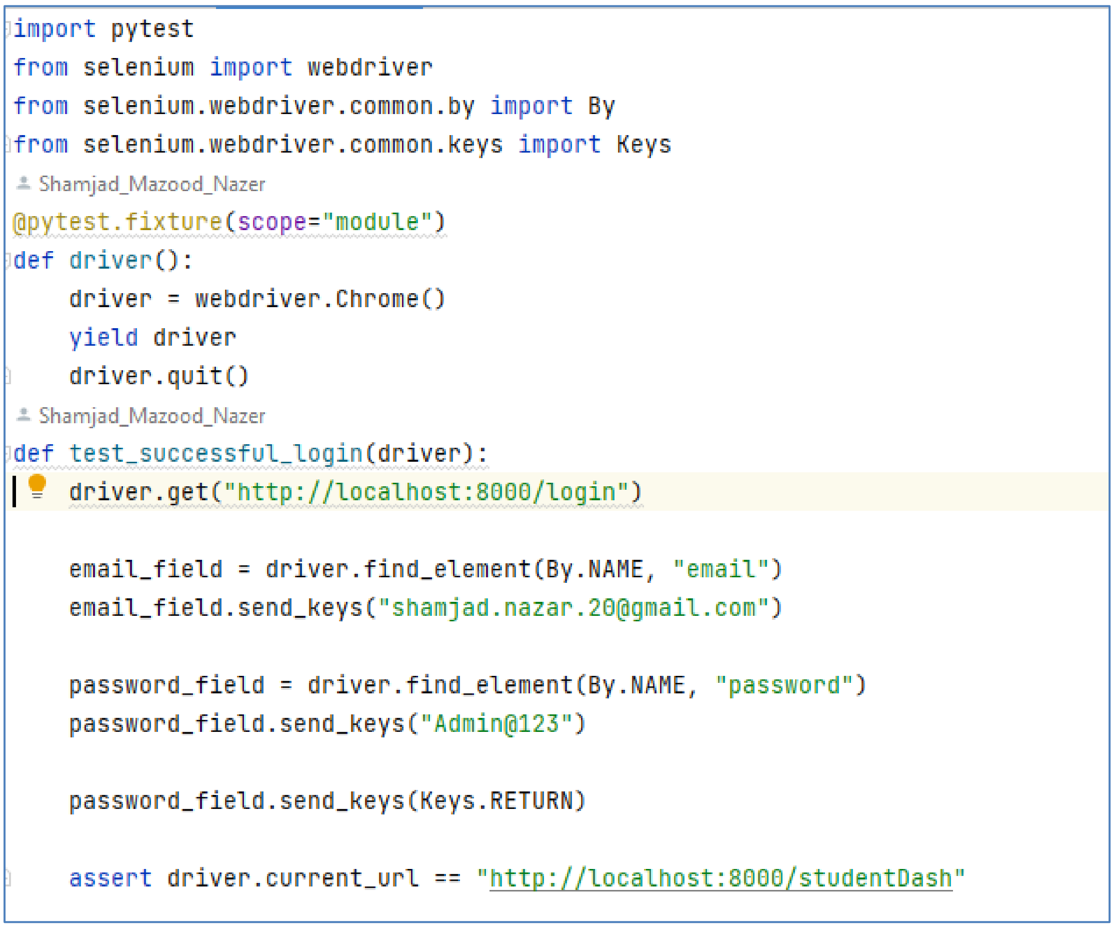
Software and other computer goods are tested automatically to make sure they abide by tight guidelines. In essence, it's a test to ensure that the hardware or software performs exactly as intended. It checks for errors, flaws, and any other problems that might occur throughout the creation of the product. 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.

5.2.6 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.

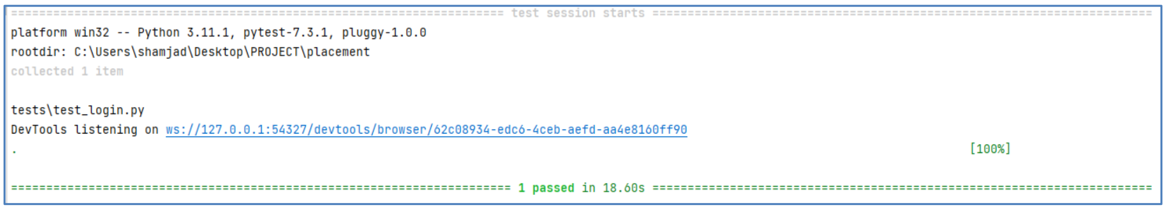
Test Case 1: Student Login

Code



```
import pytest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
Shamjad_Mazood_Nazer
@pytest.fixture(scope="module")
def driver():
    driver = webdriver.Chrome()
    yield driver
    driver.quit()
Shamjad_Mazood_Nazer
def test_successful_login(driver):
    driver.get("http://localhost:8000/login")

    email_field = driver.find_element(By.NAME, "email")
    email_field.send_keys("shamjad.nazar.20@gmail.com")

    password_field = driver.find_element(By.NAME, "password")
    password_field.send_keys("Admin@123")

    password_field.send_keys(Keys.RETURN)

    assert driver.current_url == "http://localhost:8000/studentDash"
```

Screenshot



```
====================================================================== test session starts ======================================================================
platform win32 -- Python 3.11.1, pytest-7.3.1, pluggy-1.0.0
rootdir: C:\Users\shamjad\Desktop\PROJECT\placement
collected 1 item

tests\test_login.py
DevTools listening on ws://127.0.0.1:54327/devtools/browser/62c08934-edc6-4ceb-aefd-aa4e8160ff90
.                                                                                                                                  [100%]

====================================================================== 1 passed in 18.60s ======================================================================
```

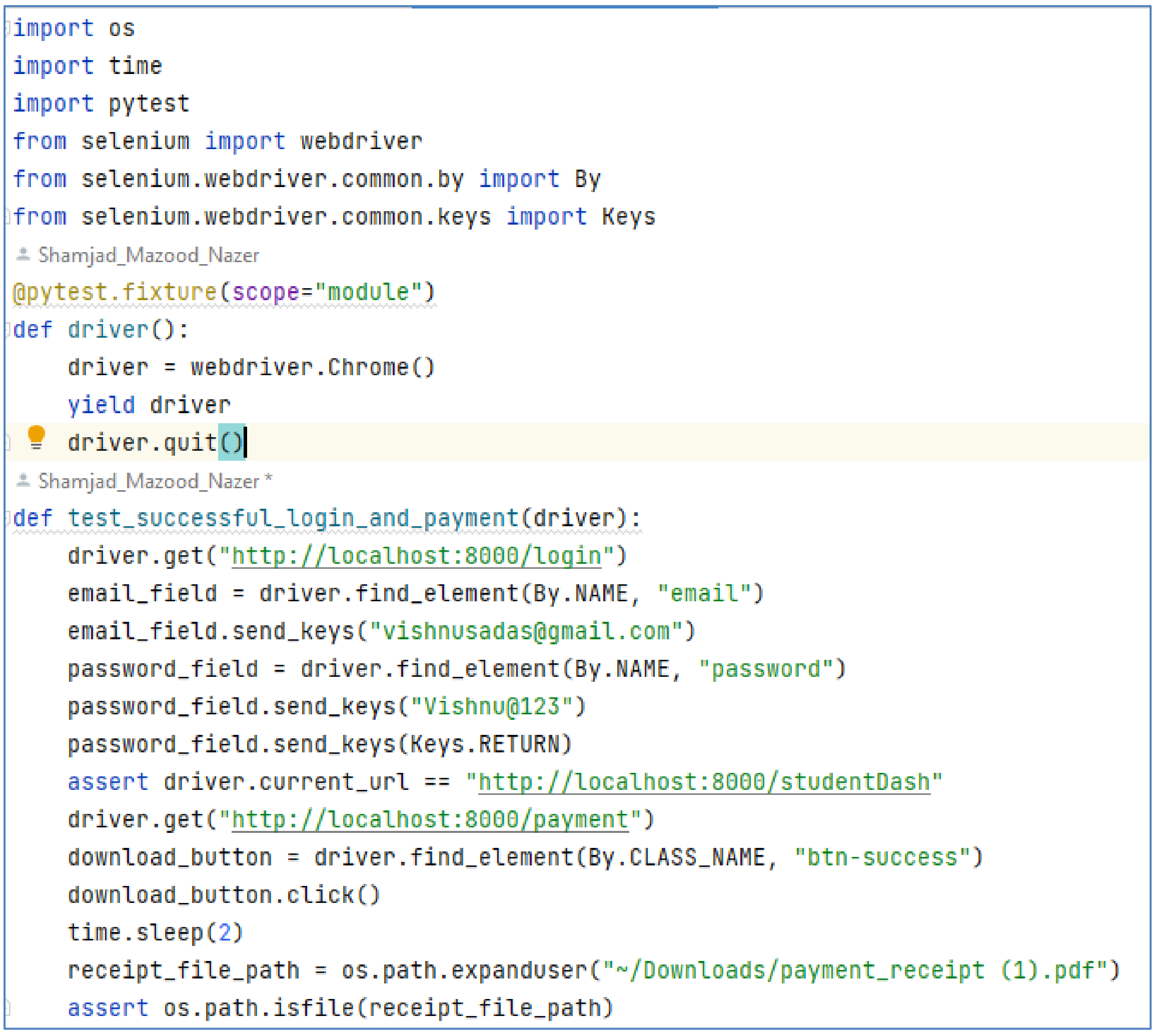
Test Report

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/ Fail)**
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	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					

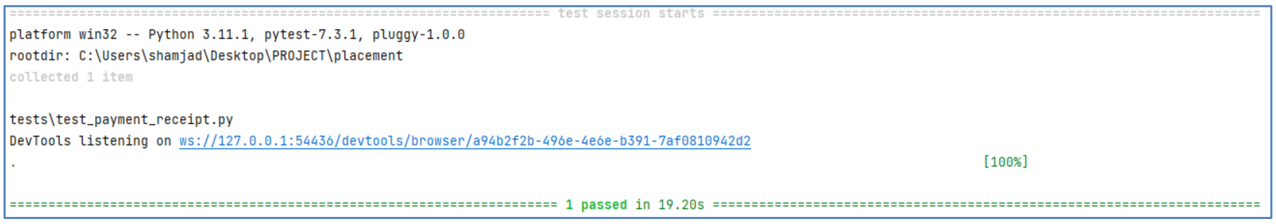
Test Case 2: Payment Receipt

Code



```
import os
import time
import pytest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
Shamjad_Mazood_Nazer
@pytest.fixture(scope="module")
def driver():
    driver = webdriver.Chrome()
    yield driver
    driver.quit()
Shamjad_Mazood_Nazer *
def test_successful_login_and_payment(driver):
    driver.get("http://localhost:8000/login")
    email_field = driver.find_element(By.NAME, "email")
    email_field.send_keys("vishnusadas@gmail.com")
    password_field = driver.find_element(By.NAME, "password")
    password_field.send_keys("Vishnu@123")
    password_field.send_keys(Keys.RETURN)
    assert driver.current_url == "http://localhost:8000/studentDash"
    driver.get("http://localhost:8000/payment")
    download_button = driver.find_element(By.CLASS_NAME, "btn-success")
    download_button.click()
    time.sleep(2)
    receipt_file_path = os.path.expanduser("~/Downloads/payment_receipt (1).pdf")
    assert os.path.isfile(receipt_file_path)
```

Screenshot



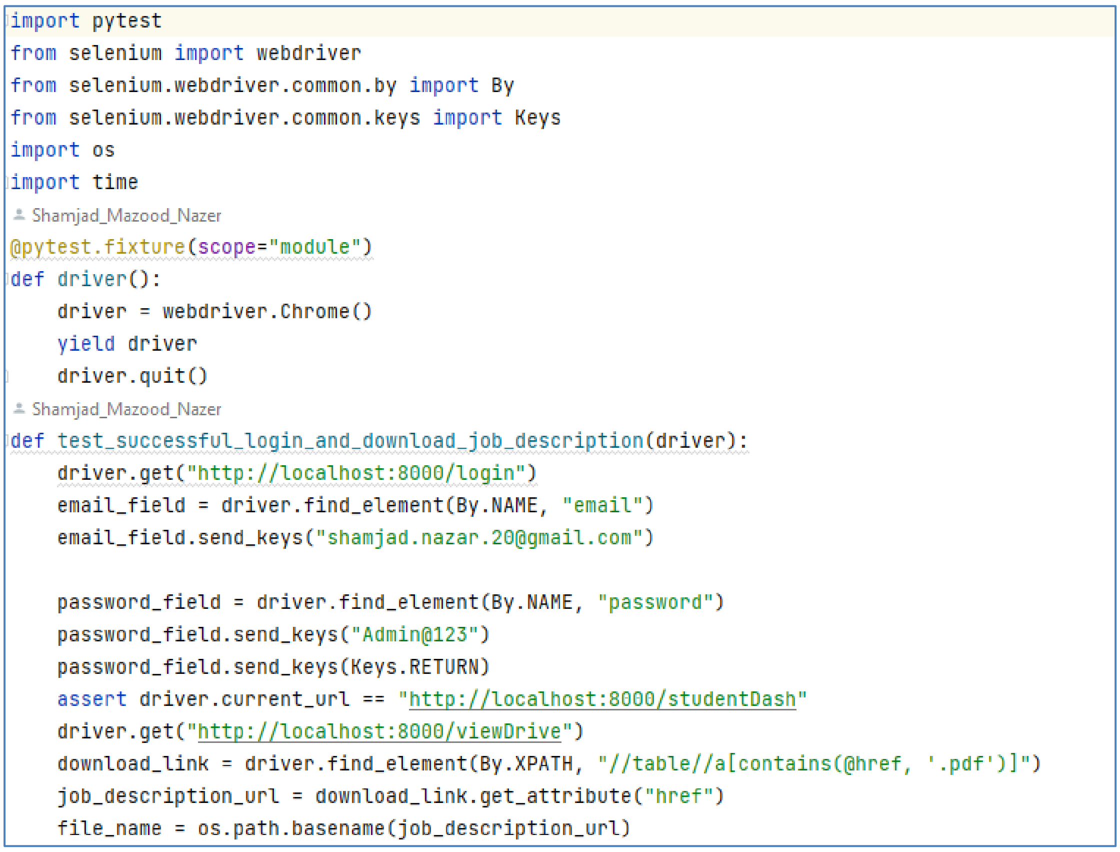
```
=============================================================== test session starts ===============================================================
platform win32 -- Python 3.11.1, pytest-7.3.1, pluggy-1.0.0
rootdir: C:\Users\shamjad\Desktop\PROJECT\placement
collected 1 item

tests\test_payment_receipt.py
DevTools listening on ws://127.0.0.1:54436/devtools/browser/a94b2f2b-496e-4e6e-b391-7af0810942d2
.                                                                                                                        [100%]

=============================================================== 1 passed in 19.20s ================================================================
```

Test report

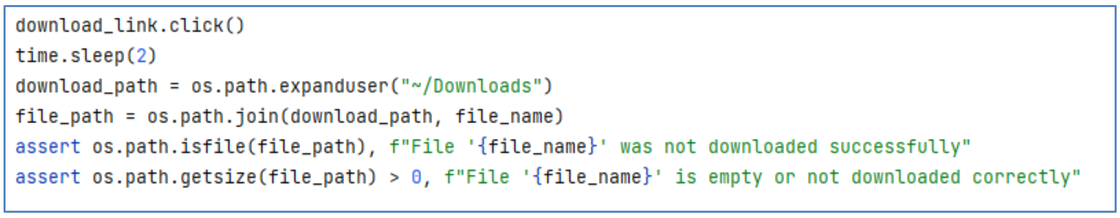
Test Case 2					
Project Name: Zyphor					
Payment Receipt Test Case					
Test Case ID: Test_2			**Test Designed By:** Shamjad Mazood Nazer		
Test Priority(Low/Medium/High): Medium			**Test Designed Date:** 10-05-2023		
Module Name: Payment			**Test Executed By :** Mr. Rony Tom		
Test Title : Download 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/ Fail)**
1	Accessing the login page		Login page is to be visible	Login page appeared	Pass
2	Give a real email	Email: 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

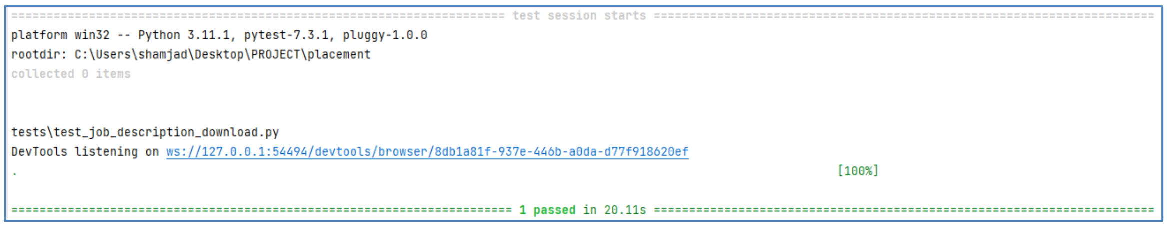


```
import pytest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import os
import time
Shamjad_Mazood_Nazer
@pytest.fixture(scope="module")
def driver():
    driver = webdriver.Chrome()
    yield driver
    driver.quit()
Shamjad_Mazood_Nazer
def test_successful_login_and_download_job_description(driver):
    driver.get("http://localhost:8000/login")
    email_field = driver.find_element(By.NAME, "email")
    email_field.send_keys("shamjad.nazar.20@gmail.com")

    password_field = driver.find_element(By.NAME, "password")
    password_field.send_keys("Admin@123")
    password_field.send_keys(Keys.RETURN)
    assert driver.current_url == "http://localhost:8000/studentDash"
    driver.get("http://localhost:8000/viewDrive")
    download_link = driver.find_element(By.XPATH, "//table//a[contains(@href, '.pdf')]")
    job_description_url = download_link.get_attribute("href")
    file_name = os.path.basename(job_description_url)
```



```
download_link.click()
time.sleep(2)
download_path = os.path.expanduser("~/Downloads")
file_path = os.path.join(download_path, file_name)
assert os.path.isfile(file_path), f"File '{file_name}' was not downloaded successfully"
assert os.path.getsize(file_path) > 0, f"File '{file_name}' is empty or not downloaded correctly"
```

Screenshot



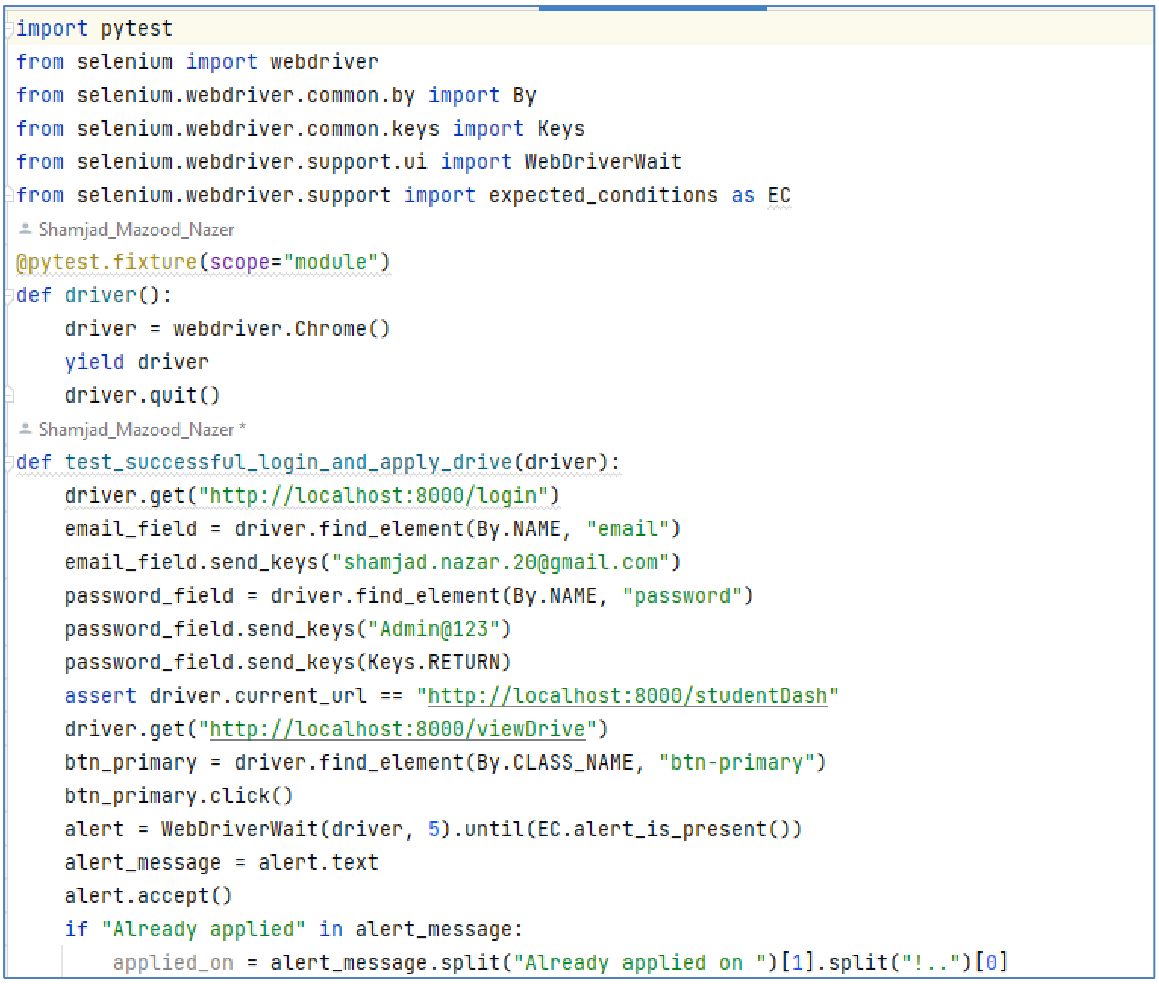
```
======================================================================= test session starts ========================================================================
platform win32 -- Python 3.11.1, pytest-7.3.1, pluggy-1.0.0
rootdir: C:\Users\shamjad\Desktop\PROJECT\placement
collected 0 items


tests\test_job_description_download.py
DevTools listening on ws://127.0.0.1:54494/devtools/browser/8db1a81f-937e-446b-a0da-d77f918620ef
.                                                                                                                        [100%]

======================================================================== 1 passed in 20.11s ========================================================================
```

Test report

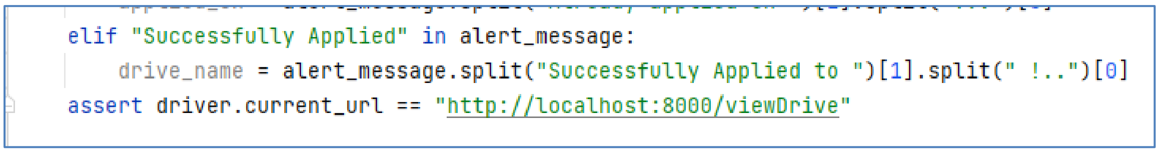
Project Name: Zyphor					
Download Job Description File Test Case					
Test Case ID: Test_3			**Test Designed By:** Shamjad Mazood Nazer		
Test Priority(Low/Medium/High): Medium			**Test Designed Date:** 10-05-2023		
Module Name: Placement Drives			**Test Executed By :** Mr. Rony Tom		
Test Title : Student 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/ Fail)**
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	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

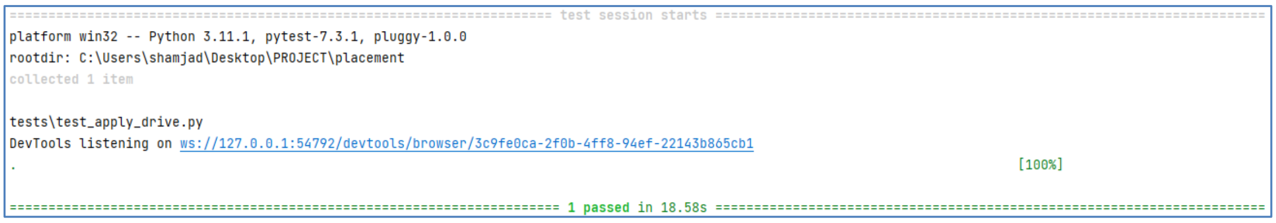


```
import pytest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
 Shamjad_Mazood_Nazer
@pytest.fixture(scope="module")
def driver():
    driver = webdriver.Chrome()
    yield driver
    driver.quit()
 Shamjad_Mazood_Nazer *
def test_successful_login_and_apply_drive(driver):
    driver.get("http://localhost:8000/login")
    email_field = driver.find_element(By.NAME, "email")
    email_field.send_keys("shamjad.nazar.20@gmail.com")
    password_field = driver.find_element(By.NAME, "password")
    password_field.send_keys("Admin@123")
    password_field.send_keys(Keys.RETURN)
    assert driver.current_url == "http://localhost:8000/studentDash"
    driver.get("http://localhost:8000/viewDrive")
    btn_primary = driver.find_element(By.CLASS_NAME, "btn-primary")
    btn_primary.click()
    alert = WebDriverWait(driver, 5).until(EC.alert_is_present())
    alert_message = alert.text
    alert.accept()
    if "Already applied" in alert_message:
        applied_on = alert_message.split("Already applied on ")[1].split("!..")[0]
```



```
    elif "Successfully Applied" in alert_message:
        drive_name = alert_message.split("Successfully Applied to ")[1].split(" !..")[0]
    assert driver.current_url == "http://localhost:8000/viewDrive"
```

Screenshot



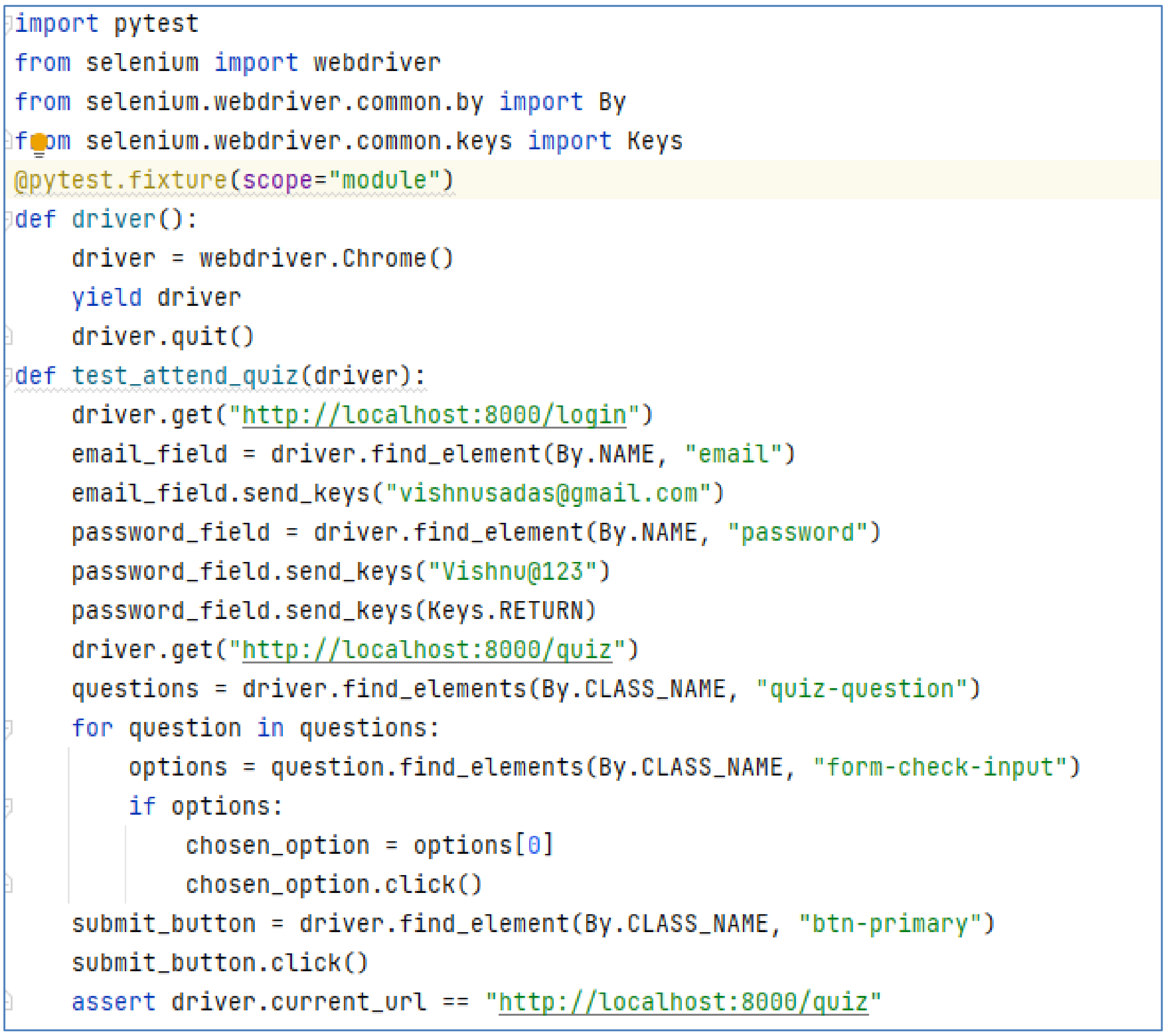
```
======================================================================== test session starts ========================================================================
platform win32 -- Python 3.11.1, pytest-7.3.1, pluggy-1.0.0
rootdir: C:\Users\shamjad\Desktop\PROJECT\placement
collected 1 item

tests\test_apply_drive.py
DevTools listening on ws://127.0.0.1:54792/devtools/browser/3c9fe0ca-2f0b-4ff8-94ef-22143b865cb1
.                                                                                                                                [100%]

========================================================================= 1 passed in 18.58s =========================================================================
```

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/ Fail)**
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	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					

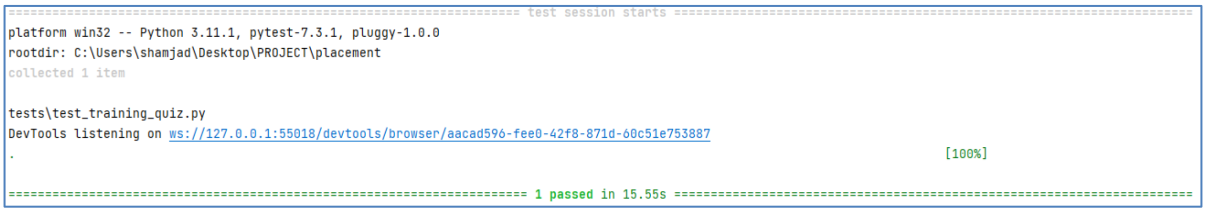
Test Case 5:

Code



```python
import pytest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
@pytest.fixture(scope="module")
def driver():
    driver = webdriver.Chrome()
    yield driver
    driver.quit()
def test_attend_quiz(driver):
    driver.get("http://localhost:8000/login")
    email_field = driver.find_element(By.NAME, "email")
    email_field.send_keys("vishnusadas@gmail.com")
    password_field = driver.find_element(By.NAME, "password")
    password_field.send_keys("Vishnu@123")
    password_field.send_keys(Keys.RETURN)
    driver.get("http://localhost:8000/quiz")
    questions = driver.find_elements(By.CLASS_NAME, "quiz-question")
    for question in questions:
        options = question.find_elements(By.CLASS_NAME, "form-check-input")
        if options:
            chosen_option = options[0]
            chosen_option.click()
    submit_button = driver.find_element(By.CLASS_NAME, "btn-primary")
    submit_button.click()
    assert driver.current_url == "http://localhost:8000/quiz"
```

Screenshot



```
======================================================================= test session starts =======================================================================
platform win32 -- Python 3.11.1, pytest-7.3.1, pluggy-1.0.0
rootdir: C:\Users\shamjad\Desktop\PROJECT\placement
collected 1 item

tests\test_training_quiz.py
DevTools listening on ws://127.0.0.1:55018/devtools/browser/aacad596-fee0-42f8-871d-60c51e753887
.                                                                                                                                 [100%]

======================================================================== 1 passed in 15.55s ========================================================================
```

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/ Fail)**
1	Accessing the login page		Login page is to be visible	Login page appeared	Pass
2	Give a real email	Email: 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

6.1INTRODUCTION

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.

6.2 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.

6.2.1 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.

6.2.2 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.

6.2.3 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".

6.2.4 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

Screenshot

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 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.

7.2 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
- www.jquery.com
- http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf
- www.agilemodeling.com/artifacts/useCaseDiagram.html
- www.agilemodeling.com/
- www.tutorialspoint.org
- https://getbootstrap.com/docs/4.0/getting-started/introduction/
- https://stackoverflow.com/
- http://www.geeksforgeeks.com/

CHAPTER 9
APPENDIX

9.1 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_o
n']
    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()['transactio
n_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()['transactio
n_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

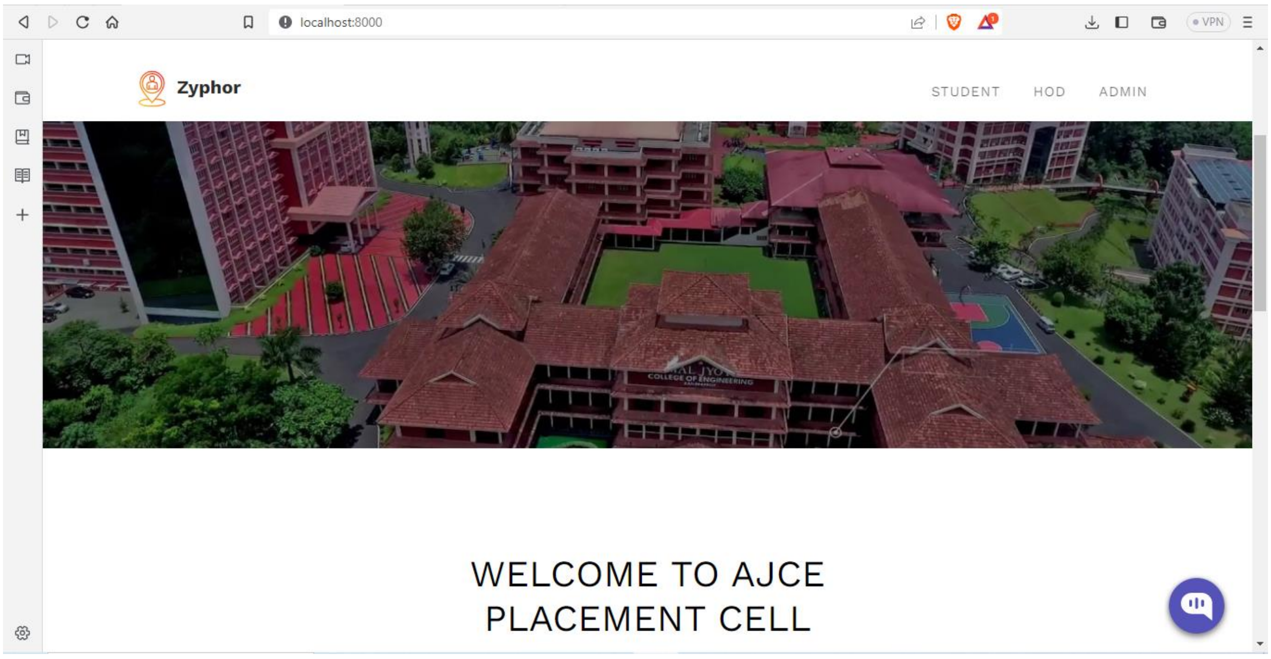
```
@admin.register(AikenFile)
class AikenFileAdmin(admin.ModelAdmin):
    list_display = ['id', 'name', 'uploaded_on', 'time', 'start_date',
'end_date', 'file']
    ordering = ['id']

    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):
                    return
                answer_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

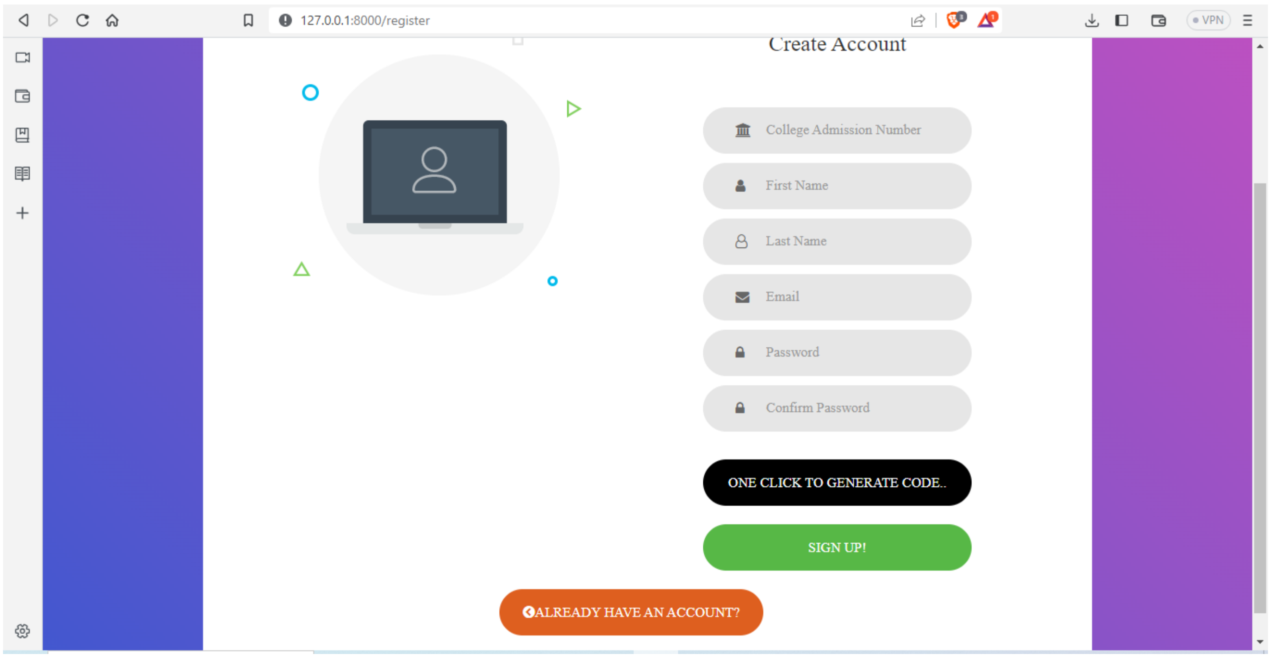
```
def export_to_csv(modeladmin, request, queryset):
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="students
data.csv"'

    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"
```

9.2 Screen Shots

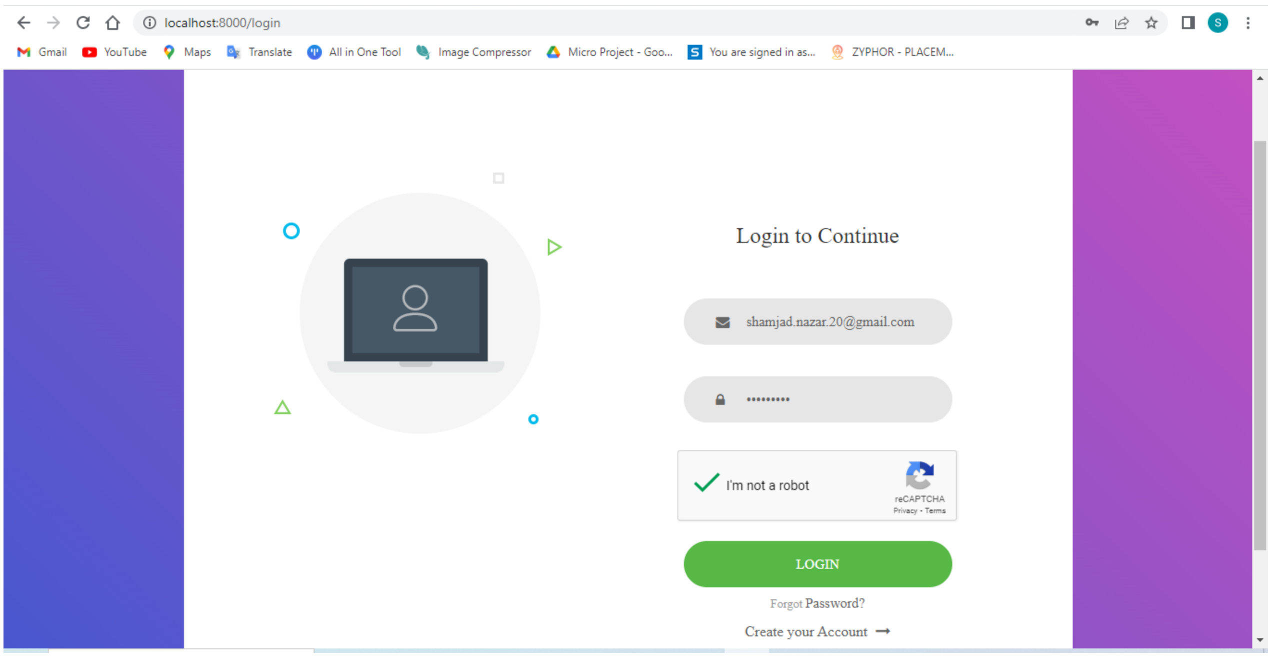
Home



Registration



Login



localhost:8000/login

Gmail YouTube Maps Translate All in One Tool Image Compressor Micro Project - Goo... You are signed in as... ZYPHOR - PLACEM...

Login to Continue

shamjad.nazar.20@gmail.com

•••••••••

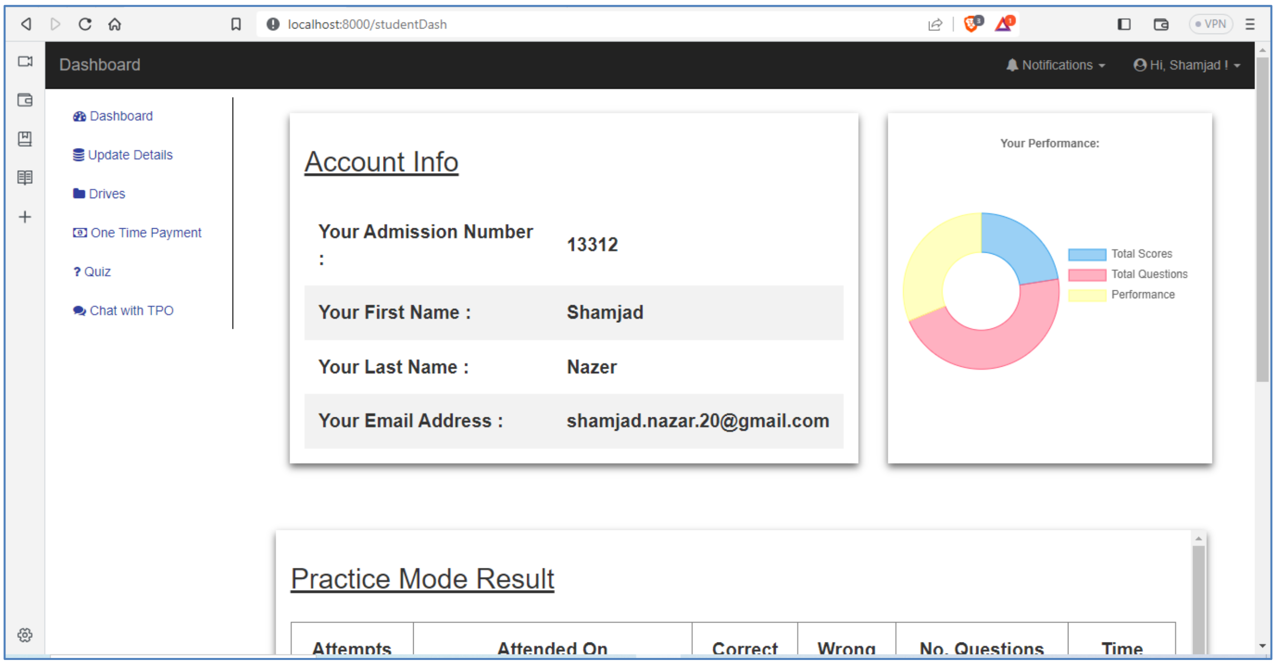
I'm not a robot

LOGIN

Forgot Password?

Create your Account ⟶

Student Dashboard



localhost:8000/studentDash

Dashboard

Notifications Hi, Shamjad !

- Dashboard
- Update Details
- Drives
- One Time Payment
- Quiz
- Chat with TPO

Account Info

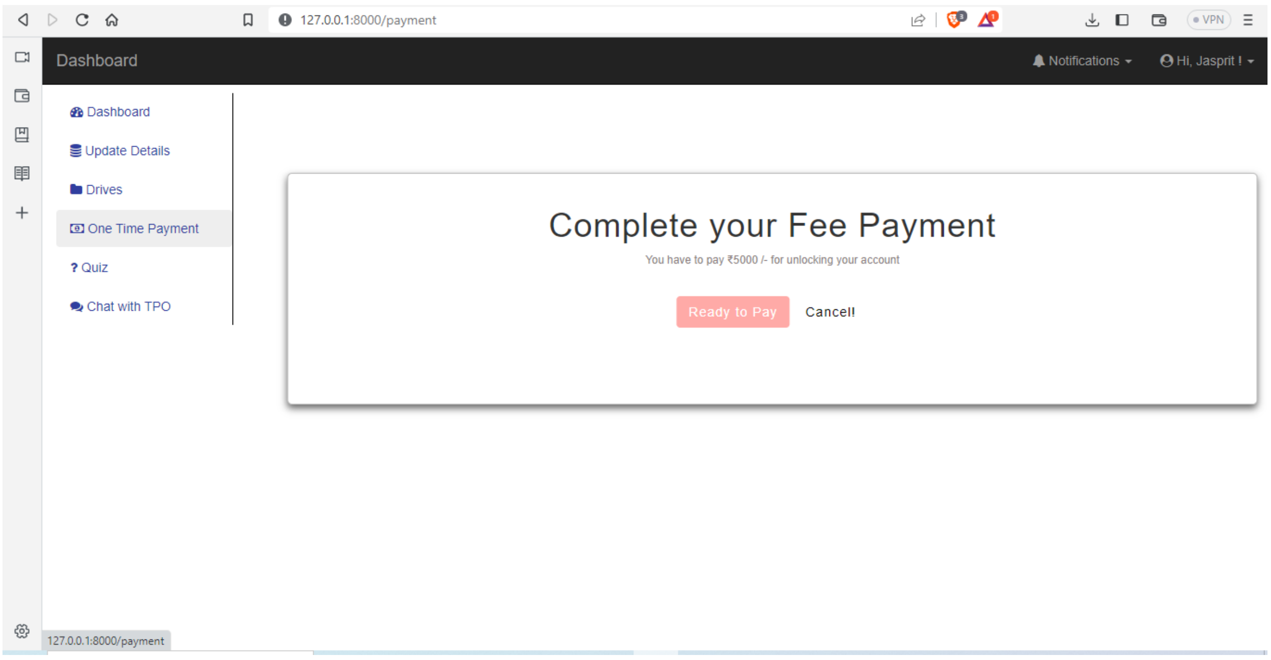
Your Admission Number :	13312
Your First Name :	Shamjad
Your Last Name :	Nazer
Your Email Address :	shamjad.nazar.20@gmail.com

Your Performance:

Practice Mode Result

Attempts	Attended On	Correct	Wrong	No. Questions	Time

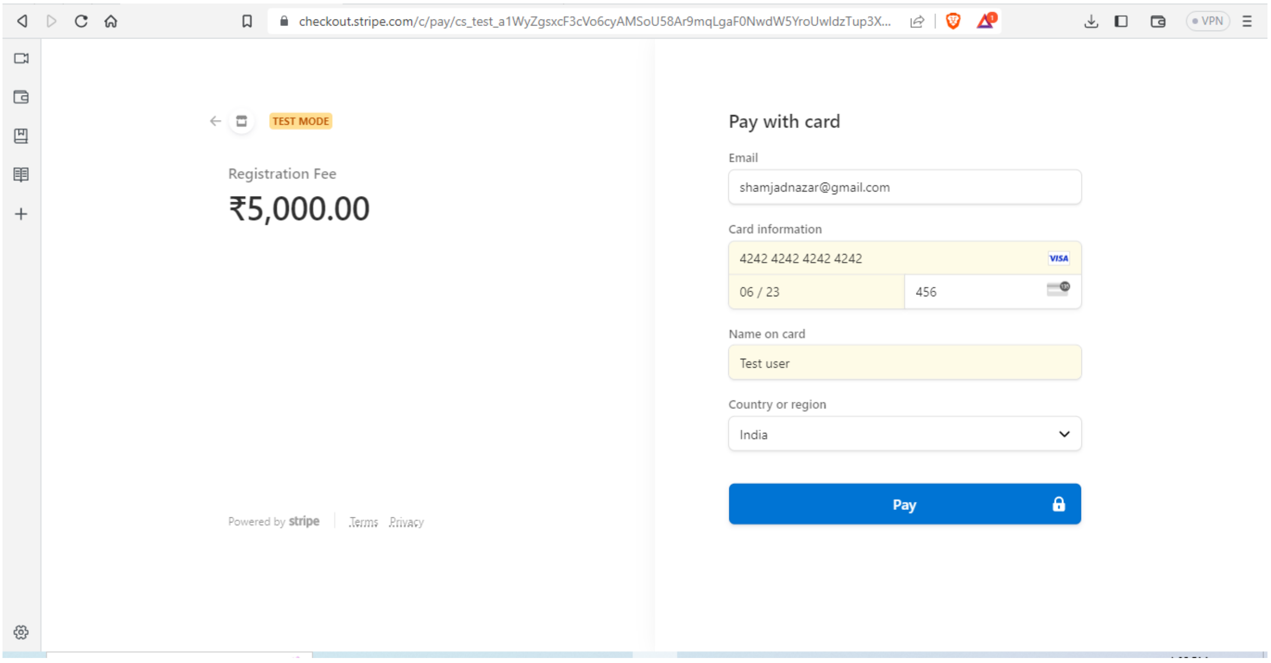
Payment



Complete your Fee Payment

You have to pay ₹5000 /- for unlocking your account

Stripe API Payment


checkout.stripe.com/c/pay/cs_test_a1WyZgsxcF3cVo6cyAMSoU58Ar9mqLgaF0NwdW5YroUwldzTup3X...
TEST MODE
Powered by stripe
Terms
Privacy
Pay

Registration Fee

₹5,000.00

Pay with card

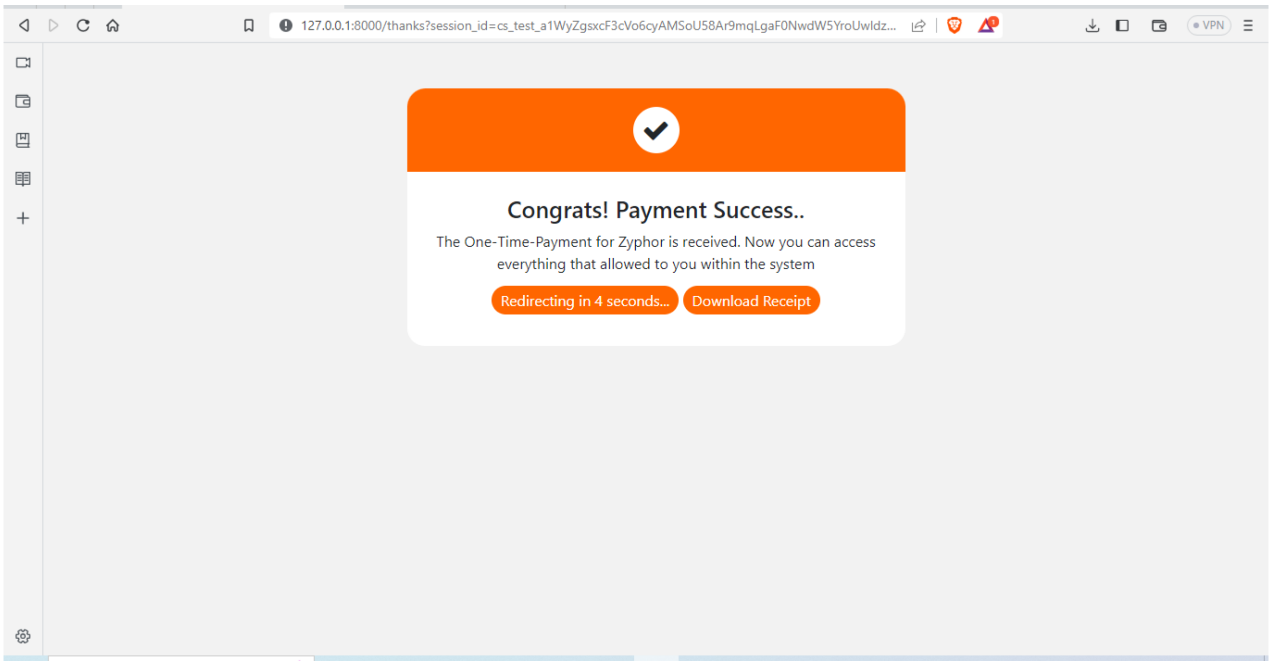
Email: shamjadnazar@gmail.com

Card information: 4242 4242 4242 4242, 06 / 23, 456

Name on card: Test user

Country or region: India

Payment Successful

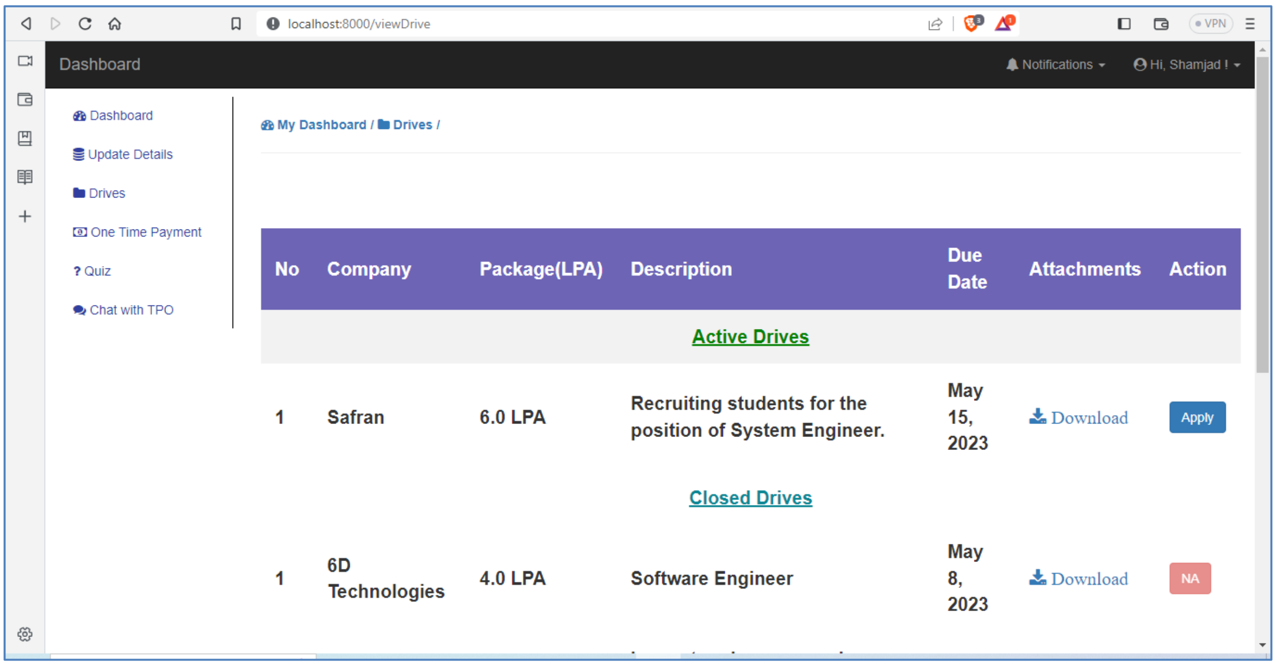


127.0.0.1:8000/thanks?session_id=cs_test_a1WyZgsxcF3cVo6cyAMSoU58Ar9mqLgaF0NwdW5YroUwldz...

Congrats! Payment Success..

The One-Time-Payment for Zyphor is received. Now you can access everything that allowed to you within the system

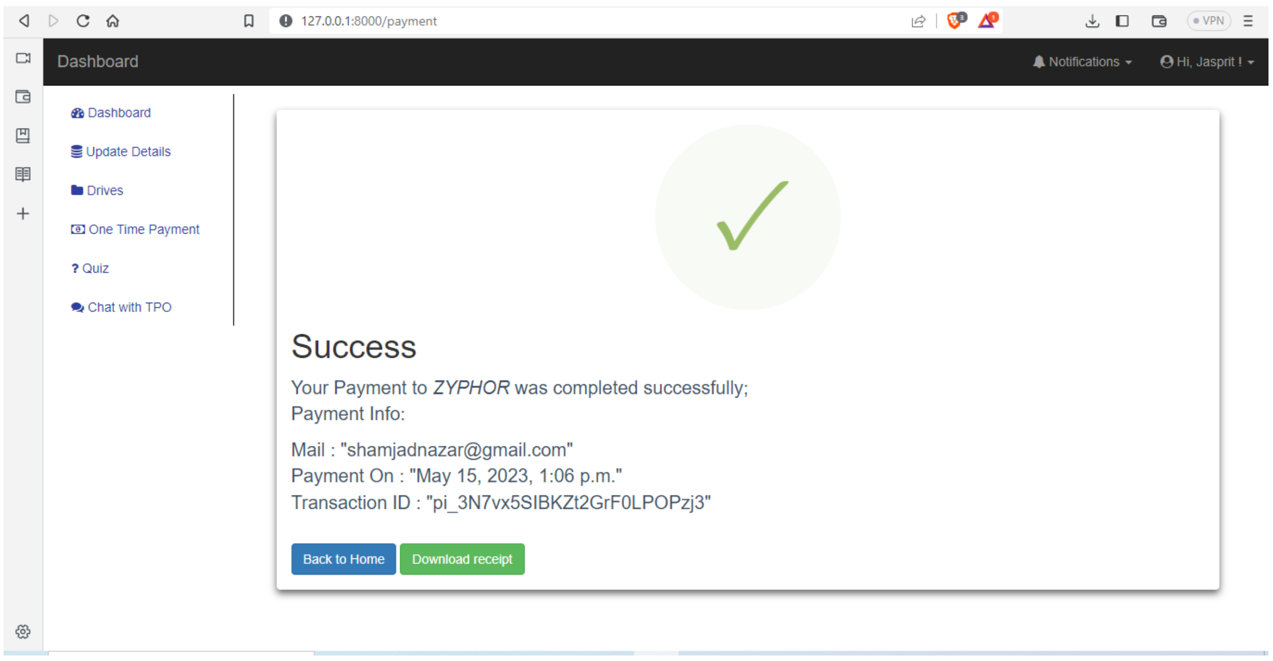
List of Drives



localhost:8000/viewDrive

No	Company	Package(LPA)	Description	Due Date	Attachments	Action
			Active Drives			
1	Safran	6.0 LPA	Recruiting students for the position of System Engineer.	May 15, 2023	Download	Apply
			Closed Drives			
1	6D Technologies	4.0 LPA	Software Engineer	May 8, 2023	Download	NA

Payment Success

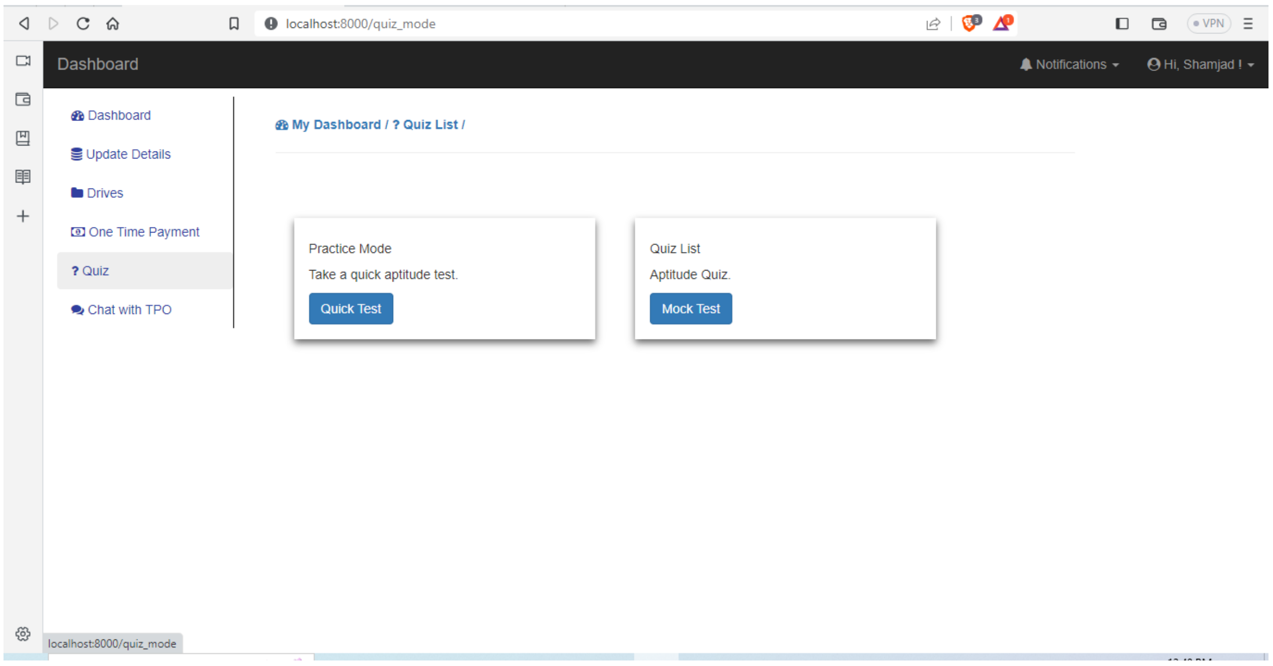


Success

Your Payment to *ZYPHOR* was completed successfully;
Payment Info:

Mail : "shamjadnazar@gmail.com"
Payment On : "May 15, 2023, 1:06 p.m."
Transaction ID : "pi_3N7vx5SIBKZt2GrF0LPOPzj3"

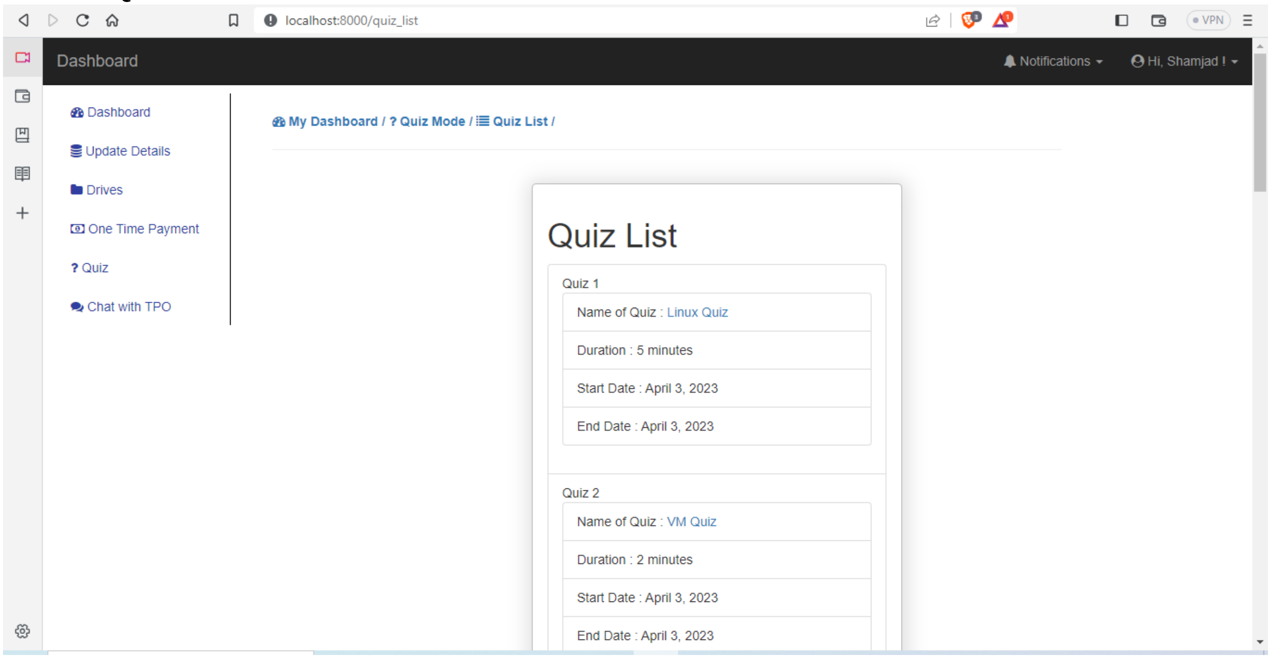
Quiz Mode



Practice Mode
Take a quick aptitude test.

Quiz List
Aptitude Quiz.

Aiken Quiz List


localhost:8000/quiz_list

Dashboard

Notifications · Hi, Shamjad !

Dashboard
Update Details
Drives
One Time Payment
? Quiz
Chat with TPO

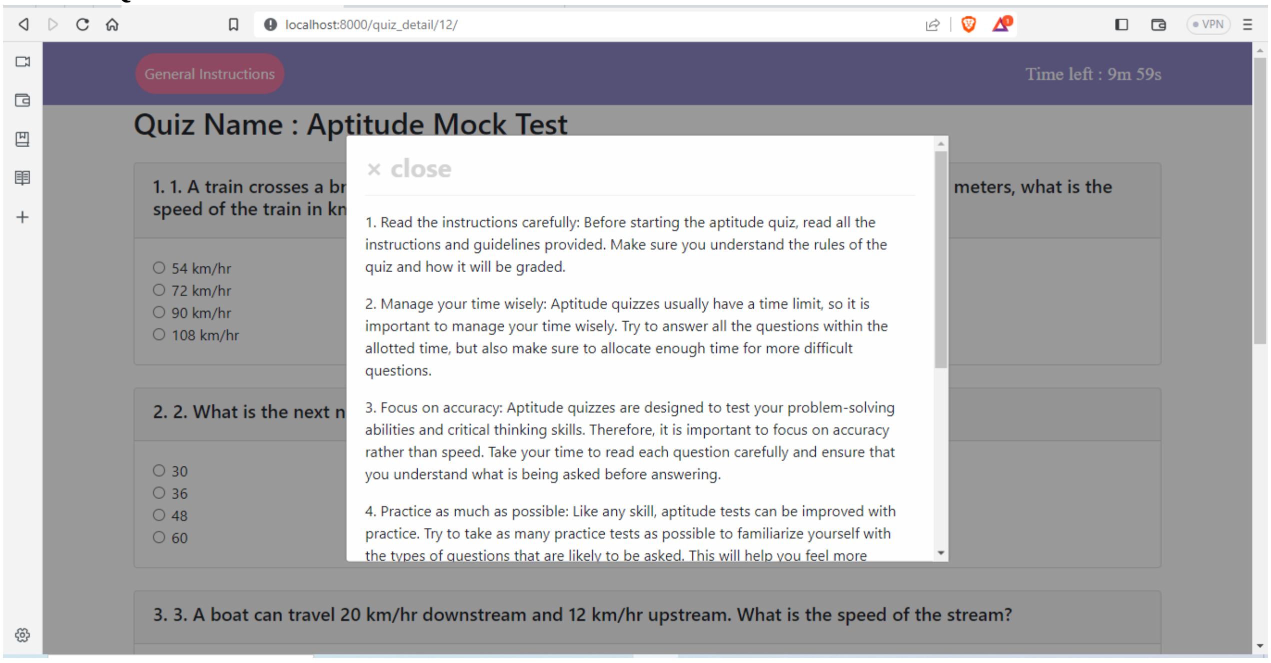
My Dashboard / ? Quiz Mode / Quiz List /

Quiz List

Quiz 1
Name of Quiz : Linux Quiz
Duration : 5 minutes
Start Date : April 3, 2023
End Date : April 3, 2023

Quiz 2
Name of Quiz : VM Quiz
Duration : 2 minutes
Start Date : April 3, 2023
End Date : April 3, 2023

Aiken Quiz Interface


localhost:8000/quiz_detail/12/

General Instructions

Time left : 9m 59s

Quiz Name : Aptitude Mock Test

1. 1. A train crosses a br … meters, what is the speed of the train in kn

○ 54 km/hr
○ 72 km/hr
○ 90 km/hr
○ 108 km/hr

2. 2. What is the next n

○ 30
○ 36
○ 48
○ 60

3. 3. A boat can travel 20 km/hr downstream and 12 km/hr upstream. What is the speed of the stream?

× close

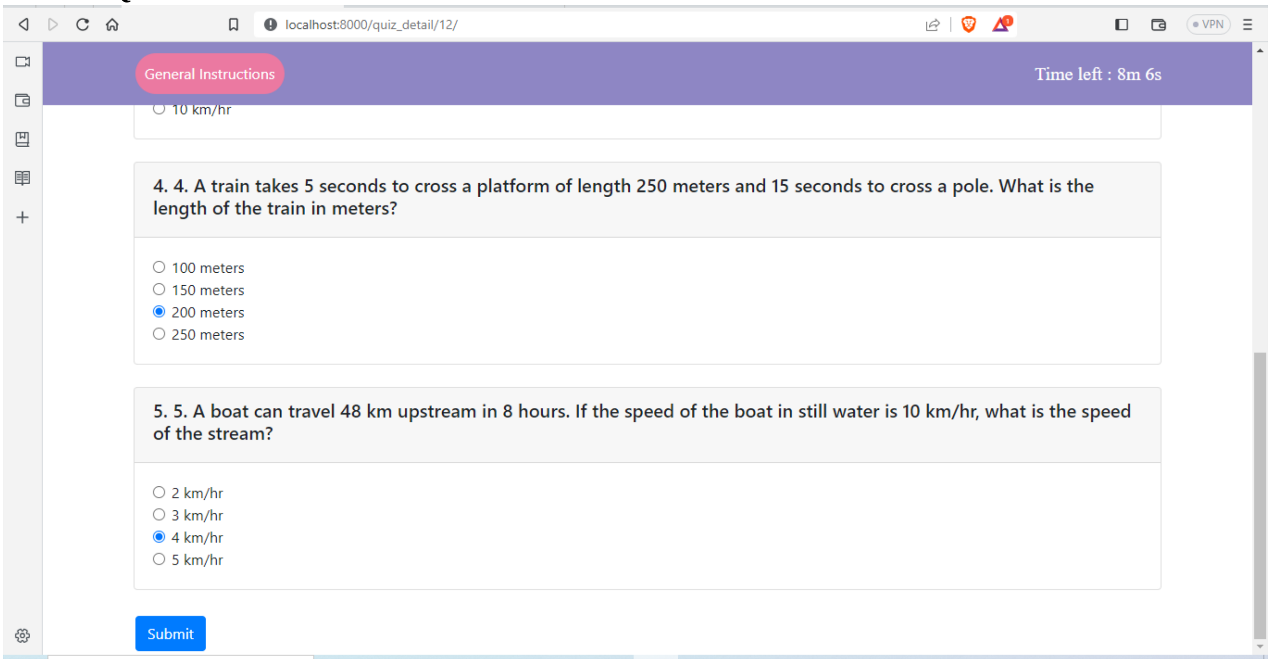
1. Read the instructions carefully: Before starting the aptitude quiz, read all the instructions and guidelines provided. Make sure you understand the rules of the quiz and how it will be graded.

2. Manage your time wisely: Aptitude quizzes usually have a time limit, so it is important to manage your time wisely. Try to answer all the questions within the allotted time, but also make sure to allocate enough time for more difficult questions.

3. Focus on accuracy: Aptitude quizzes are designed to test your problem-solving abilities and critical thinking skills. Therefore, it is important to focus on accuracy rather than speed. Take your time to read each question carefully and ensure that you understand what is being asked before answering.

4. Practice as much as possible: Like any skill, aptitude tests can be improved with practice. Try to take as many practice tests as possible to familiarize yourself with the types of questions that are likely to be asked. This will help you feel more

Aiken Quiz Submit



○ 10 km/hr

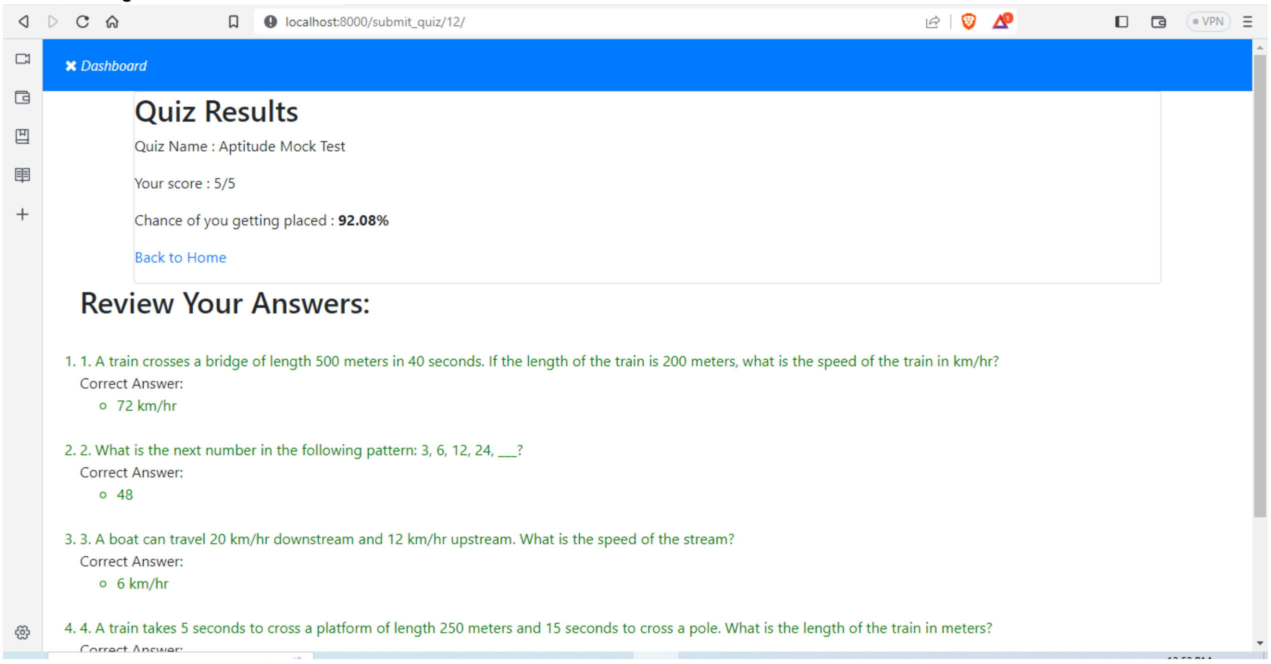
4. 4. A train takes 5 seconds to cross a platform of length 250 meters and 15 seconds to cross a pole. What is the length of the train in meters?

○ 100 meters
○ 150 meters
● 200 meters
○ 250 meters

5. 5. A boat can travel 48 km upstream in 8 hours. If the speed of the boat in still water is 10 km/hr, what is the speed of the stream?

○ 2 km/hr
○ 3 km/hr
● 4 km/hr
○ 5 km/hr

Aiken Quiz Result



Quiz Results

Quiz Name : Aptitude Mock Test

Your score : 5/5

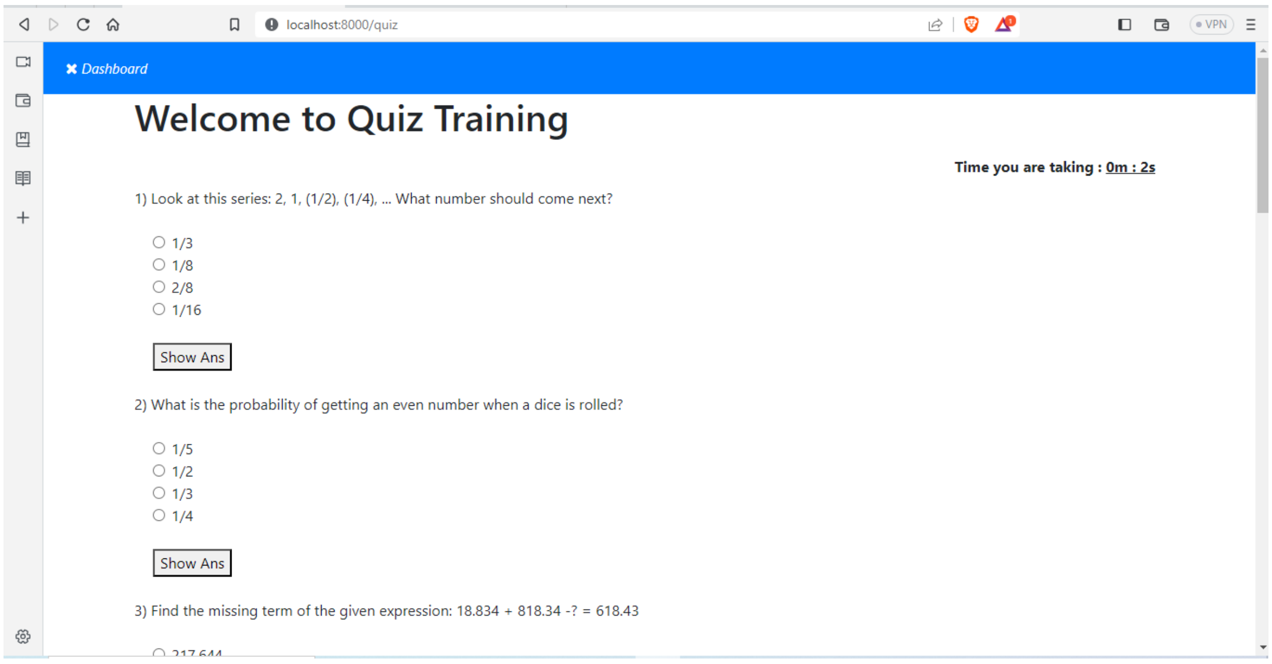
Chance of you getting placed : **92.08%**

Back to Home

Review Your Answers:

1. 1. A train crosses a bridge of length 500 meters in 40 seconds. If the length of the train is 200 meters, what is the speed of the train in km/hr?
   Correct Answer:
   - 72 km/hr
2. 2. What is the next number in the following pattern: 3, 6, 12, 24, ___?
   Correct Answer:
   - 48
3. 3. A boat can travel 20 km/hr downstream and 12 km/hr upstream. What is the speed of the stream?
   Correct Answer:
   - 6 km/hr
4. 4. A train takes 5 seconds to cross a platform of length 250 meters and 15 seconds to cross a pole. What is the length of the train in meters?
   Correct Answer:

Training Quiz



Welcome to Quiz Training

Time you are taking : 0m : 2s

1) Look at this series: 2, 1, (1/2), (1/4), ... What number should come next?

- 1/3
- 1/8
- 2/8
- 1/16

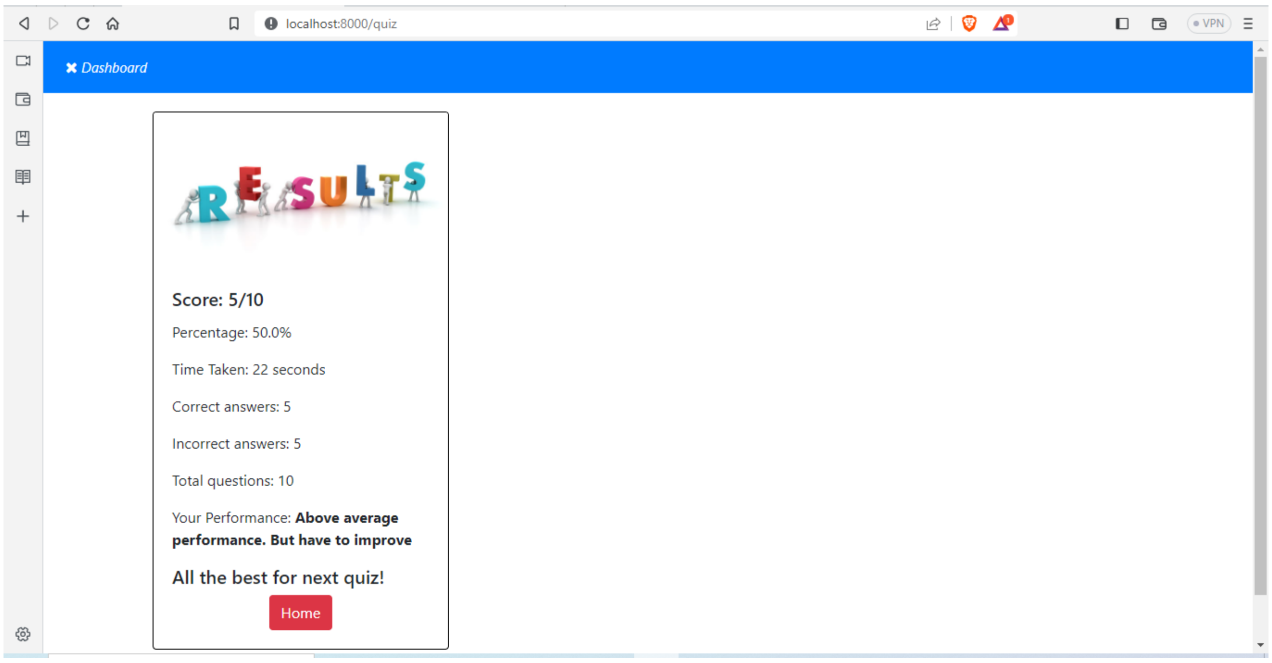
2) What is the probability of getting an even number when a dice is rolled?

- 1/5
- 1/2
- 1/3
- 1/4

3) Find the missing term of the given expression: 18.834 + 818.34 -? = 618.43

- 217.644

Training Quiz Result



Score: 5/10

Percentage: 50.0%

Time Taken: 22 seconds

Correct answers: 5

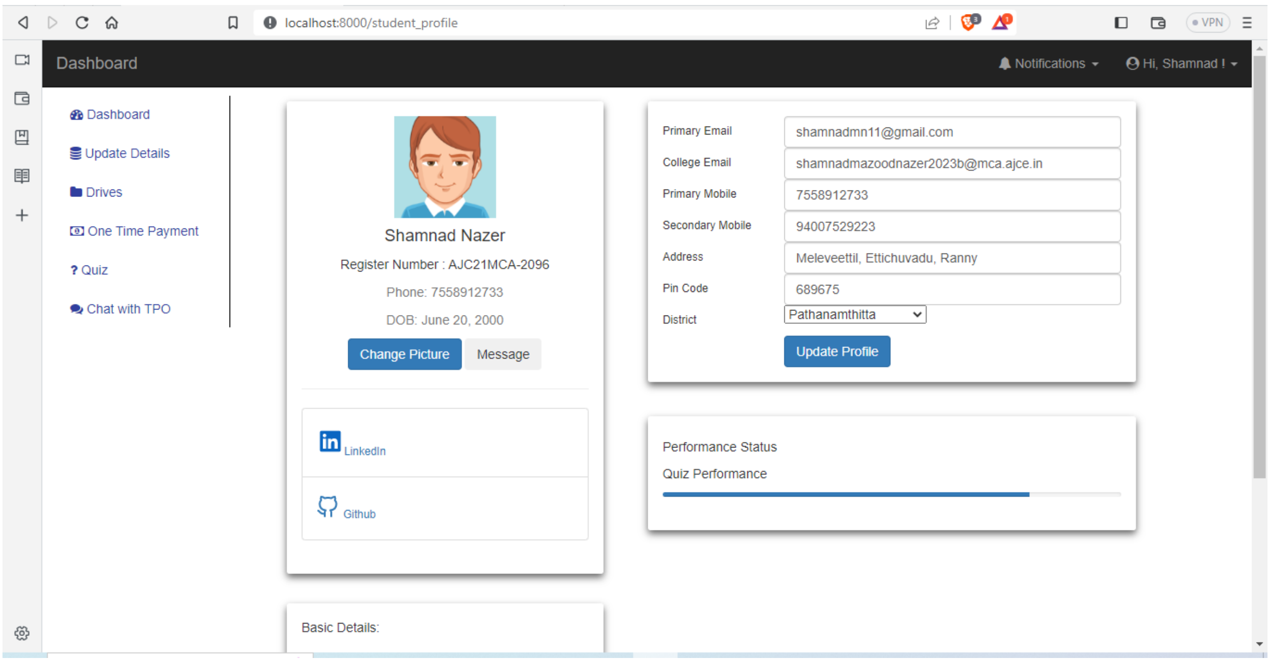
Incorrect answers: 5

Total questions: 10

Your Performance: **Above average performance. But have to improve**

All the best for next quiz!

User Profile

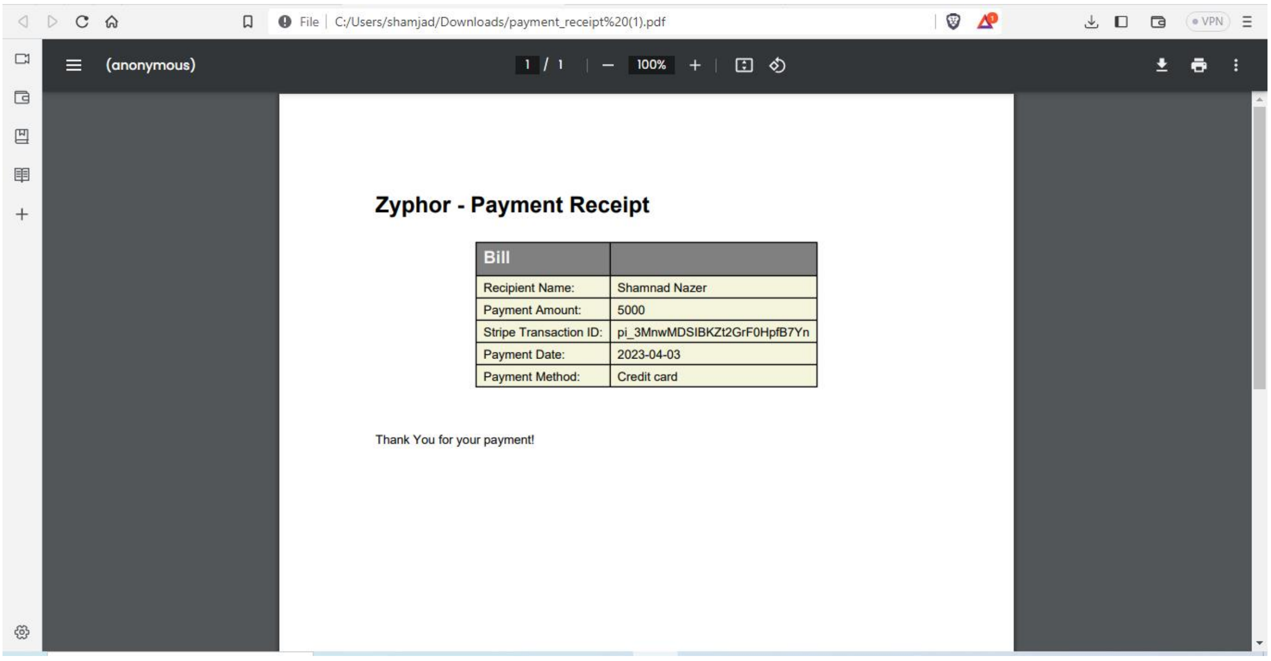


Shamnad Nazer
Register Number : AJC21MCA-2096
Phone: 7558912733
DOB: June 20, 2000

Primary Email	shamnadmn11@gmail.com
College Email	shamnadmazoodnazer2023b@mca.ajce.in
Primary Mobile	7558912733
Secondary Mobile	94007529223
Address	Meleveettil, Ettichuvadu, Ranny
Pin Code	689675
District	Pathanamthitta

Performance Status
Quiz Performance

Payment Receipt



Zyphor - Payment Receipt

Bill	
Recipient Name:	Shamnad Nazer
Payment Amount:	5000
Stripe Transaction ID:	pi_3MnwMDSIBKZt2GrF0HpfB7Yn
Payment Date:	2023-04-03
Payment Method:	Credit card

Thank You for your payment!

All screenshots

Attach Plagiarism Report