A Project Report

on

Online Examination System

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by

K.Sumanth	(17071A12E1)
N.Sai Deepak	(17071A12F5)
S.Navatej Aditya	(17071A12H0)
M.Sai Rama Krishna	(17071A1234)

DEPARTMENT OF INFORMATION TECHNOLOGY

VNR Vignana Jyothi Institute of Engineering & Technology

(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA)

Bachupally, Nizampet (S.O.) Hyderabad- 500 090.

June 2021

A Project Report

on

Online Examination System

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

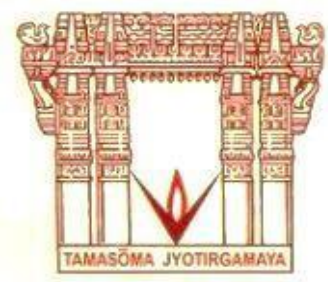
in

INFORMATION TECHNOLOGY

Submitted by

K.Sumanth	(17071A12E1)
N.Sai Deepak	(17071A12F5)
S.Navatej Aditya	(17071A12H0)
M.Sai Rama Krishna	(17071A1234)

Under the esteemed guidance of



PROJECT GUIDE

S.Rama Subba Reddy
Assistant Professor,
Dept. of Information Technology,
VNR VJIET.

DEPARTMENT OF INFORMATION TECHNOLOGY

VNR Vignana Jyothi Institute of Engineering & Technology

(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA) Bachupally, Nizampet (S.O.) Hyderabad- 500 090.

June 2021

VNR Vignana Jyothi Institute of Engineering & Technology

(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA)
Bachupally, Nizampet (S.O.) Hyderabad- 500 090

DEPARTMENT OF INFORMATION TECHNOLOGY

Date: 21 June 2021

CERTIFICATE

This is to certify that the project work entitled **"Online Examination System"** is being submitted by **K.Sumanth (17071A12E1), N.Sai Deepak (17071A12F5), S.Navatej Aditya (17071A12H0), M.Sai Rama Krishna (17071A1234)** in partial fulfillment for the award of Degree of Bachelor of Technology in Information Technology to the Jawaharlal Nehru Technological University, Hyderabad during the academic year **2020-21** is a record of bonafide work carried out by him/her under our guidance and supervision.

The results embodied in this report have not been submitted by the students to any other University or Institution for the award of any degree or diploma.

Under the Guidance of
S.Rama Subba Reddy,
Assistant Professor,
IT Department.

Head of the Department
Dr.G.Suresh Reddy,
Professor & HOD,
IT Department.

External Examiner

VNR Vignana Jyothi Institute of Engineering & Technology

(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA)
Bachupally, Nizampet (S.O.) Hyderabad- 500 090.

DEPARTMENT OF INFORMATION TECHNOLOGY

Date: 21 June 2021

DECLARATION

I hereby declare that the project entitled **"Online Examination System"** submitted to VNR Vignana Jyothi Institute of Engineering and Technology in partial fulfillment of the requirement for the award of Bachelor of Technology in Information Technology is a bonafide report of the work carried out by us under the guidance and supervision of S. Rama Subba Reddy, Assistant Professor, Department of Information Technology, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology.

To the best of my knowledge, this has not been submitted in any form to any university or institution for the award of any degree or diploma.

Signature of the Student:

K.Sumanth	N.Sai Deepak	S.Navatej Aditya	M.Sai Rama Krishna
(17071A12E1)	(17071A12F5)	(17071A12H0)	(17071A1234)

ACKNOWLEDGMENT

We express our deep sense of gratitude to our beloved **President, Dr. D. N. Rao, VNR Vignana Jyothi Institute of Engineering & Technology** for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we record our deep sense of gratitude to our beloved **Principal, Dr. C. D. Naidu** for permitting us to carry out this project.

We express our deep sense of gratitude to our beloved professor **Dr. G. Suresh Reddy, Professor and HOD, Department of Information Technology, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad-90** for the valuable guidance and suggestions, keen interest and through encouragement extended throughout project work.

We take immense pleasure to express our deep sense of gratitude to our beloved guide **S. Rama Subba Reddy, Assistant Professor in Information Technology, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad,** for his valuable suggestions and rare insights, for the constant source of encouragement and inspiration throughout my project work.

We express our thanks to all those who contributed to the successful completion of our project work.

1. KONGARA SUMANTH _____
2. NYATHA SAI DEEPAK _____
3. SURAPANENI NAVATEJ ADITYA _____
4. MAKKENA SAI RAMA KRISHNA _____

A Project Report

on

ONLINE EXAMINATION SYSTEM

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

Submitted by
KONGARA SUMANTH 17071A12E1

Under the esteemed guidance of

PROJECT GUIDE
S.Rama Subba Reddy
Assistant Professor,
Dept. of Information Technology,
VNR VJIET.

DEPARTMENT OF INFORMATION TECHNOLOGY
VNR Vignana Jyothi Institute of Engineering & Technology
(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA) Bachupally,
Nizampet (S.O.) Hyderabad- 500 090.

June 2021

A Project Report

on

ONLINE EXAMINATION SYSTEM

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

Submitted by
NYATHA SAI DEEPAK 17071A12F5

Under the esteemed guidance of

PROJECT GUIDE
S.Rama Subba Reddy
Assistant Professor,
Dept. of Information Technology,
VNR VJIET.

DEPARTMENT OF INFORMATION TECHNOLOGY
VNR Vignana Jyothi Institute of Engineering & Technology
(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA) Bachupally,
Nizampet (S.O.) Hyderabad- 500 090.

June 2021

A Project Report

on

ONLINE EXAMINATION SYSTEM

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

Submitted by
SURAPANENI NAVATEJ ADITYA 17071A12H0

Under the esteemed guidance of

PROJECT GUIDE
S.Rama Subba Reddy
Assistant Professor,
Dept. of Information Technology,
VNR VJIET.

DEPARTMENT OF INFORMATION TECHNOLOGY
VNR Vignana Jyothi Institute of Engineering & Technology
(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA) Bachupally,
Nizampet (S.O.) Hyderabad- 500 090.

June 2021

A Project Report

on

ONLINE EXAMINATION SYSTEM

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by

MAKKENA SAI RAMA KRISHNA 17071A1234

Under the esteemed guidance of

PROJECT GUIDE

S.Rama Subba Reddy

Assistant Professor,

Dept. of Information Technology,

VNR VJIET.

DEPARTMENT OF INFORMATION TECHNOLOGY

VNR Vignana Jyothi Institute of Engineering & Technology

(Autonomous Institute, Accredited by NAAC with 'A++' grade and NBA) Bachupally,

Nizampet (S.O.) Hyderabad- 500 090.

June 2021

ABSTRACT

The Online Examination System is an exam simulator that allows you to take exams on a web based platform from any place with an internet connection. It is much more efficient compared to the physical examination tests as there is no waste of time for physical correction of the test paper and lecturers can create the exams very easily.

The main purpose of this web-based examination system is to comprehensively evaluate the student using a fully automated approach that saves time and provides correct results in a short period of time. Students can respond to papers at their leisure from any location via the internet, eliminating the need for pen and paper.

Online examination is the modern solution to most of the problems that come with the traditional pen and paper examination model.

LIST OF FIGURES

INDEX

CHAPTER 1
INTRODUCTION

Due to the huge advancement of internet technology in the past few years, online examinations have become an effective way of replacing the traditional examination system. It is known by everyone that a lot of the students try to cheat when taking the traditional examination model, which might negatively affect the legitimacy of the examination. The pen and paper method of writing an examination has been going on for many years, but it might not be very efficient due to many reasons, some of them are examination location size, lack of comfort for students, long waiting time for results, examination malpractices, high costs involving printing examination papers and sometimes human errors might also occur. The change of examination and administration process from pen and paper based to online based process demands huge change in the previously established examination procedure in institutions. Lecturers, administrative staff, examiners, IT support staff, and the students have to make themselves comfortable with the new online examination system.

Online examination is efficiently used in day to day life because it saves time and also it is the most accurate system as maximum participants are increasing in today's life. Currently, there are many different kinds of online examination systems and each one will have a particular use. As there is no one type that satisfies all kinds of requirements, many institutions opt to create their own system which will perfectly fit their current infrastructure and suit their examination model.

1.1 PURPOSE OF THE PROJECT

The primary goal of the proposed system is to understand the student proficiency in academic subjects. This system can eliminate the logistic problems and disadvantages within the traditional pen-and-paper examination mode. The online examination system follows a more direct and convenient examination approach by switching to an online platform that can allow institutions to conduct exams in multiple question formats and get instant reports of the students' performance. They can conduct exams at any place while preserving the exam's credibility and integrity. The

need for a proper online examination system is clearly demonstrated by the recent COVID-19 lockdown which made it impossible for students to attend their universities and write their exam.

1.2. EXISTING METHODOLOGIES AND ITS DISADVANTAGES

As the traditional examination system is conducted in properly established physical centers, the processing of paper based records is very slow. The process goes on for weeks or probably months and causes both physical and mental stress over the students.

It is very problematic for each and every student to reach the exam center on time, the declaration of results could also take a very long time and due to this delay many students will wait for weeks to months to pursue their future interests. In the current paper based examination system, checking the number of students registered, and verification of their details manually is very difficult. It is a major waste of money and time because it necessitates a lot of manpower. Students could suffer heavy losses due to the delay in result declaration as they will not be able to pursue higher studies or attend any competitive exams to get a job interview as they do not have their examination result. The following are some of the disadvantages of the existing traditional examination system:

- The existing system results in errors, high time consumption, not very efficient and a lot of valuable resources are wasted. Storing the paper based records is also very difficult as the quantity is always increasing but the available space for managing and storing these records is very hard to get due to high costs.
- Students must mail the help center or personally visit their university for any information about their examination. As the paper load in the examination branch is high during exam times, the paper corresponding to a particular student might be very hard to find. The need of students visiting their university everytime they need information causes time waste and it is also a huge inconvenience to the students.
- In the existing examination system, there is always a possibility of tampering with student records by the examinee. There are many cases where fake degrees were created for the students as there is no proper online verification to check the credibility of the degree, and this results in unfair practices. One person's wrongdoing could easily damage the reputation of an institution.

- Sometimes there might be a repetition of the same work over and over again. This could cause data duplications and result in a lot of money being spent by the university to buy papers. Also the management of student's records is very hard. Checking of student records through manual registers and maintaining records data causes huge time consumption.
- No students are being allowed to sit unless they are present at the exam hall. So students have to reach the center to appear for their exam. Sometimes due to other problems they may not reach the stipulated time and they could in return miss the exam because the time of their examination may have passed.
- In the existing examination system, there is always a limitation for the number of students that can sit for examination at a time, and there is also a need for an invigilator to be present on the day of the exam in order to monitor the exam center when student's start writing their exam.
- It doesn't allow the appearing candidates to settle on their subjects as per choice and check the results.

1.3. PROPOSED SYSTEM

The online examination system proposed in this study is designed using a layered architectural pattern that has four layers - presentation, core module, server, and storage service.

1. Presentation Layer: This layer represents the various ranges of devices that can be used to access the online examination system via the Internet. This can include desktop machines, laptops, and mobile devices such as smartphones and tablets.
2. Core Modules: This layer refers to the key feature of the online examination system that is split into three modules. The Administrator module allows an authorized admin to add faculty to the systems database. The faculty module allows a given faculty to add examinations for the students to take. The Student module simply provides the interface for the students to take exams and check their results.
3. Server: This refers to the layer from which the online examination system can be deployed as a web application.
4. Storage Service: This layer covers the rapid retrieval and storage of data and information using a Relational Database Management System. SQLite was the database adopted in this study and it works seamlessly with any web server.

1.4. OBJECTIVE

As the current manual examination systems have numerous disadvantages, including high time consumption, more manpower requirement to conduct exams, possibility of errors in results as they are calculated manually, there is a higher chance of losing the exam paper in the traditional system compared to online systems, and checking results should also be done manually. With the recent advancement of information technology, it is now possible to overcome the flaws present in the current examination system. The proposed examination model a database can be used to save exam information, making it easier for teachers to conduct exams as they just have to add their exams preference, and students can give exams in an automated way.

1.5. THESIS ORGANIZATION

Chapter 1: This chapter presents the basic information and introduction and the necessary technical knowledge to implement.

Chapter 2: This chapter deals with the literature survey and research needed to implement the project.

Chapter 3: This chapter deals with the challenges and issues that occurred during the implementation of the project.

Chapter 4: This chapter deals with methodology design and approach

Chapter 5: This chapter deals with the implementation of the methodology.

Chapter 6: This chapter deals with the experimental results of the model.

Chapter 7: This chapter deals with the conclusion.

Chapter 8: Future Scope the Project.

Chapter 9: References collected for the Project.

CHAPTER 2
LITERATURE SURVEY

Guzman and Conejo proposed the System of Intelligent Evaluation using Tests for Tele-education (SIETTE) in 2005, which is a web-based examination system . It's a web-based exam system that allows you to create adaptive tests based on student performance. It's used to create adaptive student self-assessment exam questions that include tips and feedback for instructional goals. SIETTE includes features such as secure login and portability. Random question selection, multiple instructor support, random distributing questions, resuming where previously stopped, and random choices distribution are some of the other capabilities of this system that are missing.

In 2010, Rashad et al. presented Exam Management System (EMS), a web-based online examination system. EMS can manage the exam by conducting examinations, collecting responses, auto-grading the submitted exams, and generating test results. Secure login, numerous teachers, and portability are all supported by EMS. Resumption capability, random question selection, random question distribution, and random choice distribution are all absent from this system.

In 2011, Arvind Singh, Niraj Shirke, and Kiran Shetty suggested a proposal that uses the online examination system concept to evaluate examinees. The exams will be completely personalised. Based on the responses of the students, this system will automatically produce the results.

In 2001, Luecht identified six main problems in conducting online examination: (1) the identity of examinee can be duped and possibility of exam paper leaks by data breach, (2) measuring the students problem solving skills, (3) applying algorithms for item selection and test creation, (4) managing and running the test result properly, (5) using high bandwidth consuming tests could be problematic for few students (6) making the web based testing interfaces user friendly. Luecht has also identified many strengths of online examination including creating and

conducting tests rapidly, flexibility to conduct tests at any time, reporting results instantly, and lesser manpower is required.

Students' reactions and happiness with online assessments were evaluated as part of a study on student opinions and satisfaction with online courses. Hale discovered in 2007 through student satisfaction surveys that the primary reason students choose online education courses is convenience. Furthermore, according to Steinman in 2007, students' perceptions of online courses might be bad if they are separated from the teacher and other students. He also stated that this can impact a student's decision to stay in a class or drop out.

In 2013, Fagbola et al. created a Computer Based Test System (CBTS). It is an online examination system developed to fix difficulties such as timing flexibility, auto sign off once the allotted session time has expired, authenticity of results, independence from other systems, and robustness. Its main design purpose is to help examination procedures and address obstacles such as test conducting, automatic marking of submitted examinations, and automatic examination result generation.

In 2006, Patterson conducted a survey for students after completing an online exam. This survey results revealed that it is possible for a significant portion of students to access the online exam without much effort. It also demonstrated that the testing technology was simple to use, with students completing the exam with ease. 87 percent of respondents agreed that online assessments for academic exams should be used in the future. Patterson also discovered that the Web-based comprehensive exam processes used allowed students to take the exam at their leisure, at any time and in any location. The opportunity to choose the time and location of the exam, according to Patterson, helped pupils to feel less stressed. He also said that the issues of maintaining security and implementing stringent processes to prevent the likelihood of collusion and cheating on this type of exam must be addressed.

CHAPTER 3

ISSUES AND CHALLENGES

First issue is the challenge of state continuity. It is caused when there is a sudden disruption of exams because of a power cut or network issue. In cases like this, it becomes a big issue for continuing the exam if the system is not designed with that in sight. There is also a possibility that a person can somehow restart the PC then act innocent when they find the questions too difficult. With the current IT literacy, there is no proper way to identify if the exam disruption was on purpose or not. The examiner could also be forced to create a new exam or conduct it again on a separate day.

Another major issue with online examinations is authentication. Without an efficient authentication system, it is possible for students to trick the authentication system and cheat during the exams. It is possible to take advantage of the weaknesses/bugs in the system if strong penetration tests aren't conducted. This can allow user impersonation and there is no way to confirm this as there'll likely be no physical authentication of candidates.

Level of IT literacy can be a serious challenge. Some of the created systems can be hard to understand as they won't be very user friendly and might require considerable practice for the user to become used to the interface. All online systems must be very easy to understand and simple so everyone can understand what needs to be done just by looking at the interface. In some cases, the user reads the instructions on the screen and they might respond according to it. If unnecessary designs have been made to the interface, users will become confused by the instructions and not understand how to respond. This will lead to lots of complaints and opportunities for exploitation of the system.

All examination systems must have proper encryption methods for question papers. If encryption levels aren't good, then it is possible for someone to breach the website and leak the exam papers leading to bad examination standards. Manipulations of exam reports and data could also be done by unauthorized access. Also, it might lead to loss of examination data.

Each and every institution will have their own standard of examination and grading. Most of the current examination systems that are available off the shelf do not support these custom standards. This will cause problems integrating such systems into the institution's examination framework. Due to this issue a lot of changes need to be made either to the system or the examination structure of the institution. Another option for institutions is to invest huge amounts of money into developing a custom made examination system that fits their structure. Even with all these options, the institute will still need a physical examination system as the current online examination systems might not support all forms of examination.

CHAPTER 4

METHODOLOGY

4.1. INTRODUCTION

The Structured Systems Analysis and Design Methodology (SSADM) was employed for the development of this project.

It's a widely used methodology for the analysis and design phases of system development. It employs a systematic method to the construction of information systems, in which the modules, steps, and tasks to be completed are specified in advance.

The “Waterfall Model” of the system development life cycle is used by SSADM, in which each phase must be completed and authorised before moving on to the next. Because its diagrammatic representation resembles a waterfall, this model is named as "Water Fall."

The waterfall model is among the first attempts to explain the software development life cycle. The project is separated into numerous phases in this approach, including requirements comprehension, analysis, design, implementation, testing, and maintenance. From concept to analysis, design, implementation, testing, installation, troubleshooting, and operation and maintenance are all steps in the development process. Each stage of development follows a tight timeline, with no phases overlapping or repeating. Before moving on to the next stage, each step must be completed. SSADM primarily employs three major techniques: logical data modelling, data flow modelling, and entity event modelling.

Logical Data Modeling (LDS)

This is the process of identification, modelling, and documenting an information system's data requirements. A logical data structure and related documentation are included in a logical data flow model. It denotes entities and their connections.

Data Flow Modeling (DFM)

The process of identification, modelling, and documenting how data flows through an information system is known as data flow analysis. A dataflow model is a collection of pre-built data flow diagrams that are accompanied by necessary documentation. Processes, data stores, external entities, and data flows are all represented in DFM.

Entity Event Modeling (EEM)

EEM is the act of identification, modeling and documentation of events that influence each entity as well as the sequence in which they occur. A set of entity life histories (one for each entity) and associated supporting documentation are present in an EEM.

SSADM provides a step-by-step flow or waterfall view of system development in light of the preceding explanation. Each subsequent step builds on the one before it. The SSADM processes/stages include feasibility analysis, requirements specifications, logical system specifications, and physical design.

4.2. PROPOSED APPROACH AND ITS FLOW DIAGRAM

In the proposed approach, there are three modules: Student module, faculty module and administrator module.

Any student can register their account with email. After email authentication is done, they can freely access the account when needed. In the main interface, students can take tests within the specified time deadlines set by the faculty and check their previous exam results.

The faculty module can only be accessed when it is authorised by the administrator during the initial registration, this is to prevent any random person from impersonating a faculty member. After authentication, it can be freely accessed when needed by the faculty. Exam questions can be created and papers can be customized. If a student is switching tabs then an alert mail will be sent to the professor's email. Marks of all the students who have taken the exam will also be visible.

The django administrator page is used by the admin to give access to an account as a faculty member, monitor student details, check and modify the question papers, and deleting accounts to prevent any unwanted access to the website. All the alert, authentication mails to faculty and students through the administrator email.

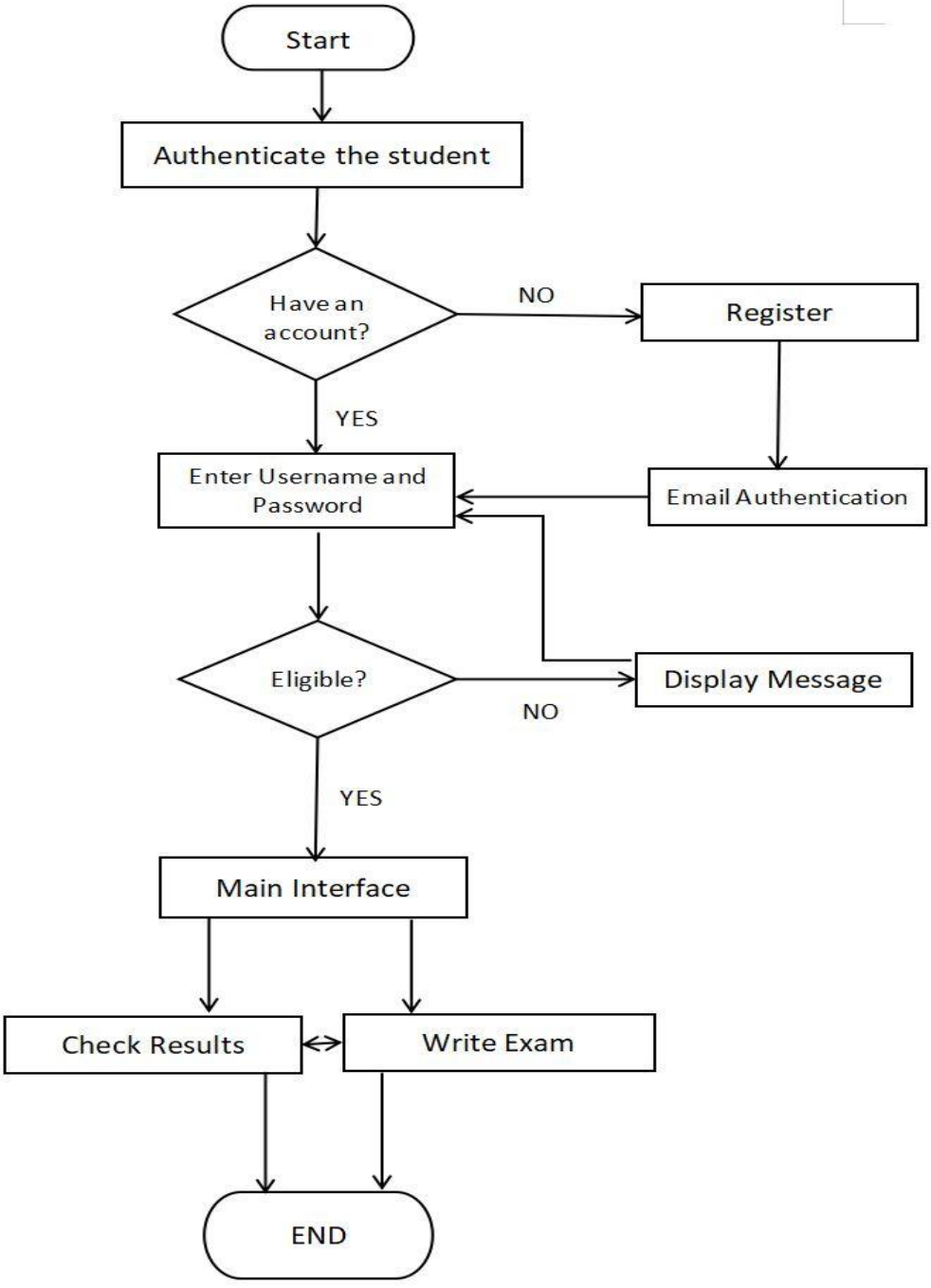


FIG 4.1: Student Module Flowchart

CHAPTER 5
IMPLEMENTATION

5.1. FUNCTIONAL REQUIREMENT ANALYSIS

Unlike traditional examination modes, the proposed online examination system can dynamically handle various examination information, such as time limitations, automatic test result calculation, and so on. Students can use a browser to access the web examination system from anywhere, giving it a high degree of freedom.

The major purpose of the web examination system is to learn how to use it to automatically conduct exams. After the exam, the system can read the paper according to a specific strategy based on the reference answer. To reach this goal, we must manage examination questions methodically and effectively, as well as automatically provide timely examination results.

The following is an analysis of the system's functional requirements:

(1) User Login

- Student login: Register by entering the student name, password and complete email authentication, log in using the registered details to enter the home page.
- Faculty login: Enter the login username and password, if it is authorized by the administrator, you will be logged in to the faculty home page. Examination question bank management and checking student marks is possible.
- Administrator login: Enter the username and password. After correctly logging in, you can enter the backend controlling page.

(2) Backend Website Management

- User management: Manage student, teacher, and administrative information, including operations such as adding, editing, deleting, and querying.
- Examination paper management: Examination papers can be viewed, deleted or new ones can be added. New questions can also be added or deleted.

(3) Examination System

- Online examination: First the students have to log in and if there is no error then they enter the examination page, the system will automatically show all the available tests to the student, after the student opens the exam, the countdown timer will be displayed. Confirm and finish the exam by the end of the test deadline, otherwise the system will force the exam to end if the timer runs out. The result will be calculated at the end of the exam and will be displayed to both student and professor.
- Cheating Protection: An email will be sent to the professor if the student switches tabs continuously.
- Result query: After the exam, the result will be sent to both student and professor. It can be accessed through the results menu.

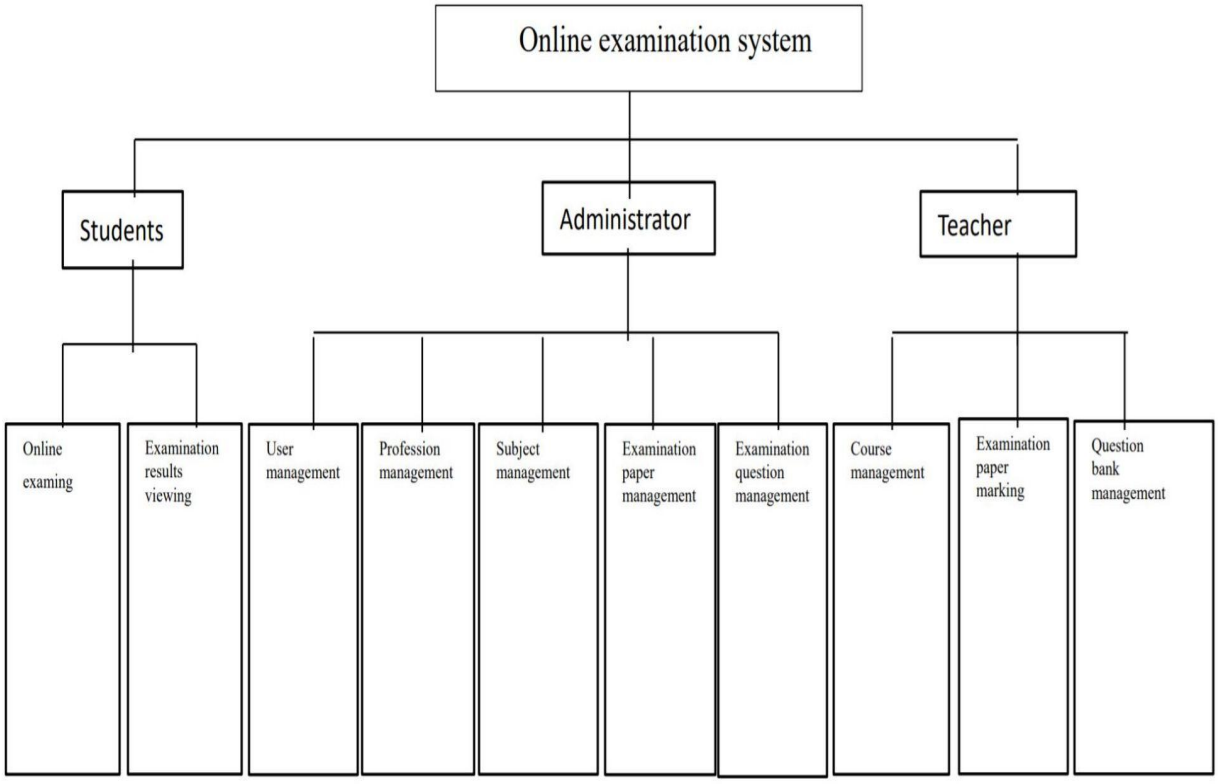


FIG 5.1: FUNCTIONAL REQUIREMENTS

5.2. SYSTEM DESIGN

System design is the process of determining the hardware or software architecture, components, modules, interface, and data for a computing system in order to meet specific requirements. It's possible to think of it as a computerised variation of system theory.

5.2.1. Architecture Design

The web-based examination system is developed on the python django web framework and the front end is developed with HTML and Javascript. SQLite database is used.

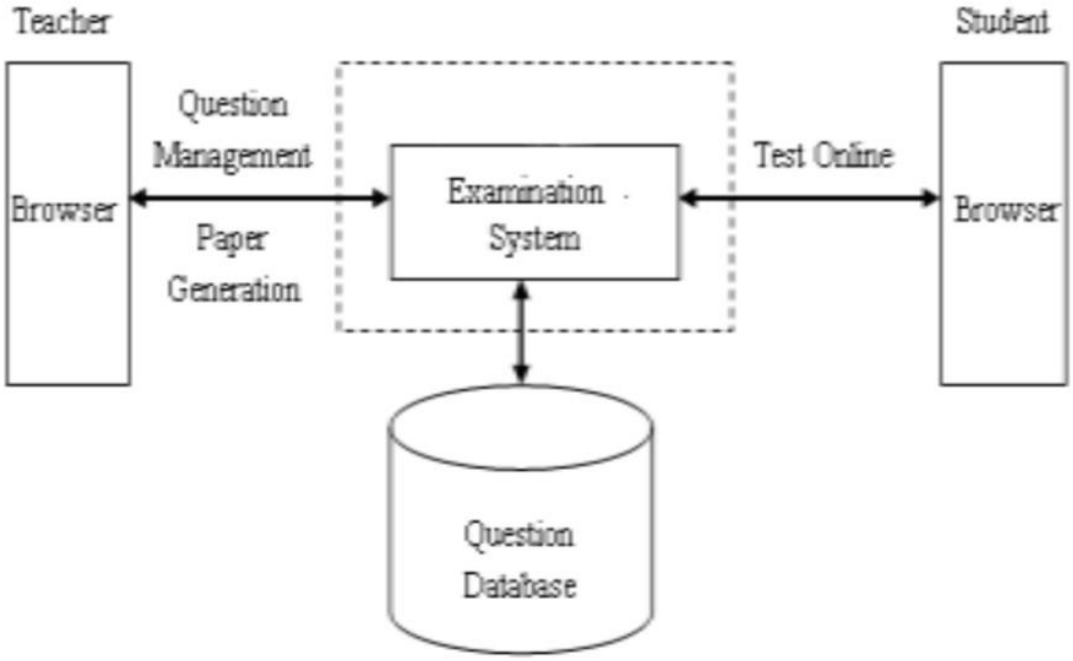


FIG 5.2: System Architecture

5.2.2. Data Storage Structure Design

- The administrator table contains an admin name, admin password and the email.
- Professor information table contains professor details and login information.
- The student information table contains student personal details, results and login information.

- The question paper table includes the subject name, examination paper name, question numbers, and answer key.
- Answer number, test number, student number, correct answer, and answer marks are all listed in the answer table.
- The student's name, subject name, question paper, and total score are all stored in the result table.

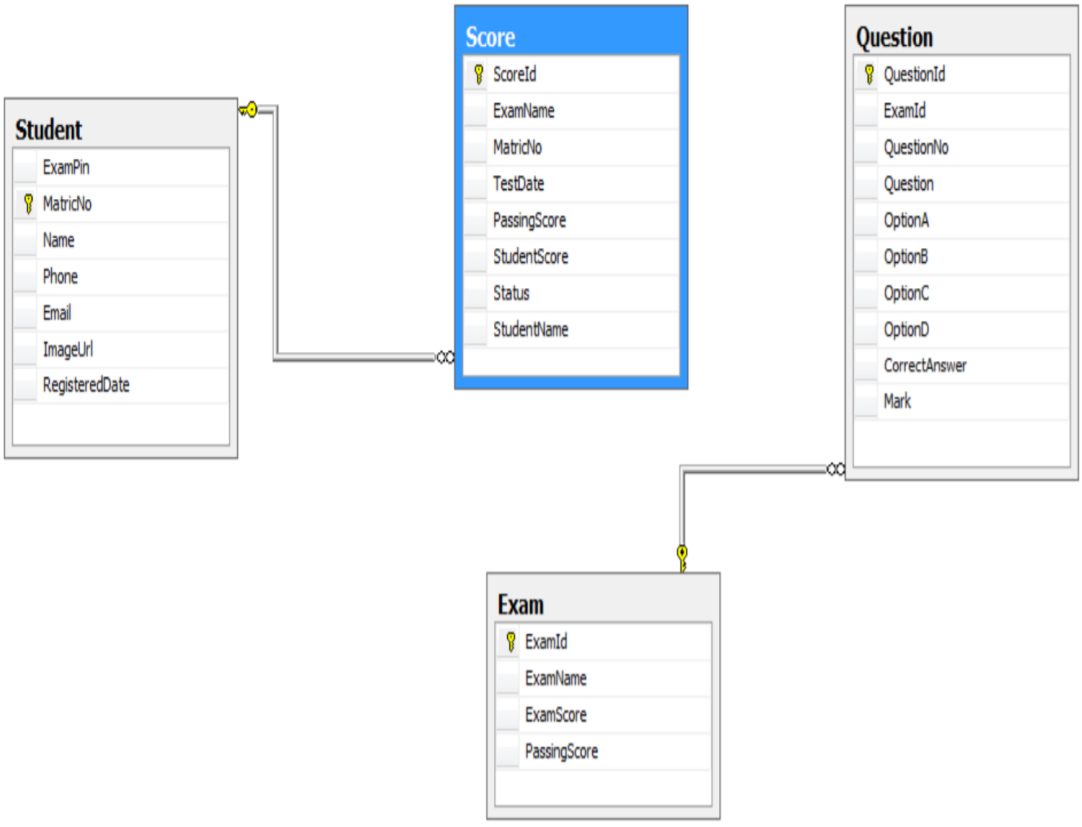


FIG 5.3: DATABASE DESIGN

5.3. ENVIRONMENT

Python is an interpreted and a high-level programming language. Python has simple English statements which enable us to understand the code without much prior knowledge. Python's design makes code readability with its use of significant whitespace. It also follows indentation which doesn't hinder much in analysing or debugging the code. It follows an object-oriented approach to have users a clear view of the logical and functional part of the projects. The projects need not be a small scale.

5.3.1. Interface - Jupyter Notebook, Anaconda Prompt

The Jupyter Notebook is a freely downloadable web application. It's an open source programme that allows us to create and share documents including code, text, or equations.. Jupyter is a project that is used to develop the Jupyter user interface to support multiple programming languages.

Jupyter Notebooks is a subproject from a project called Ipython. It mainly supports Julia, Python and R programming languages and hence the name Jupyter. Jupyter along with the Ipython kernel. It enables us to write your code in Python, R. There are around 100 other kernels which are available in Jupyter that can be used to run the program.

The notebook extends helps to compute or visualize data for building an application which requires analysis of data using a console-based approach. Jupyter also has extra features of documenting and transfer of information which is useful to maintain records on a large scale project. As every other platform it supports development, execution and computation of code in an IDE. The Jupyter notebook integrates two main components while developing any application or platform. These include:

- Web application: It is a web-based tool for interactive documents production which incorporates explanatory text, mathematics, computations and their rich media.
- Notebook documents: It includes all of the web application's accessible content, including as computational inputs and outputs, explanatory language, mathematics, pictures, and rich media representations of things.

5.3.2. Python Packages

Django: It is a high-level Python web application framework that focuses on fast development and a straightforward, practical design. It was created by experienced developers to handle the bulk of the issues that come with Web development, allowing you to concentrate on your project instead of wasting time on simple things. It's open source and free.

Asgiref: ASGI is an asynchronous successor to WSGI that allows asynchronous Python servers and web applications to interact.
Few ASGI base libraries included in this package are:

- Sync-to-async and async-to-sync function wrappers, asgiref.sync
- Server basic classes, asgiref.server
- A WSGI-to-ASGI adapter, in asgiref.wsgi

Bcrypt: It's a popular password storage algorithm that was created with long-term password storage in mind. Django does not use it by default because it needs the usage of third-party libraries, but because most developers might want to use it, Django supports it without any complications.

Pillow: The Python Imaging Library enhances the image processing capabilities of your Python interpreter. This library can handle a large variety of file types, has a very quick and dynamic representation, and has rather powerful image processing capabilities.

The primary purpose of the image library is to enable rapid access to data saved in a few common pixel formats. It should be an excellent starting point for any image processing software.

Pytz: It makes the Olson tz database available in Python. Using Python 2.4 or higher, this module enables for precise and cross-platform timezone computations. It also eliminates the problem of daylight savings time by automatically adjusting time.

Six: It's a compatibility library for Python 2 and 3. It supplies utility functions for easing over Python version differences with the purpose of developing Python code that works on both Python versions.

Sqlparse: It's a Python SQL parser that doesn't validate the data. Parsing, dividing, and formatting SQL statements are all supported. Python 3.5+ is required to use the module.

Validate_email is a package for Python that checks if an email is valid, properly formatted and really exists.

CHAPTER 6

RESULTS

In this section, you will come across the outputs that are obtained by the implementation of the proposed system.

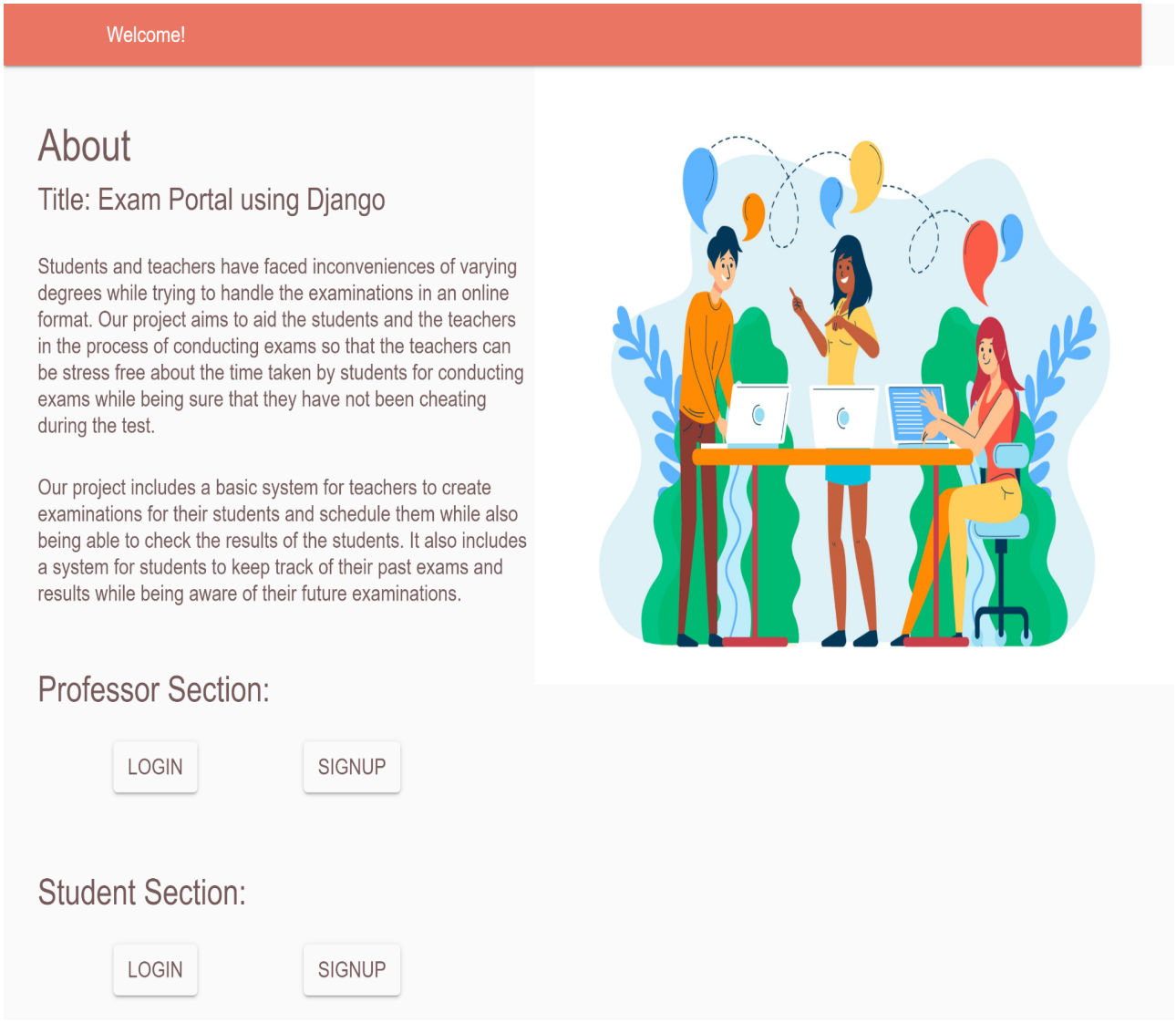


FIG 6.1: Home Page

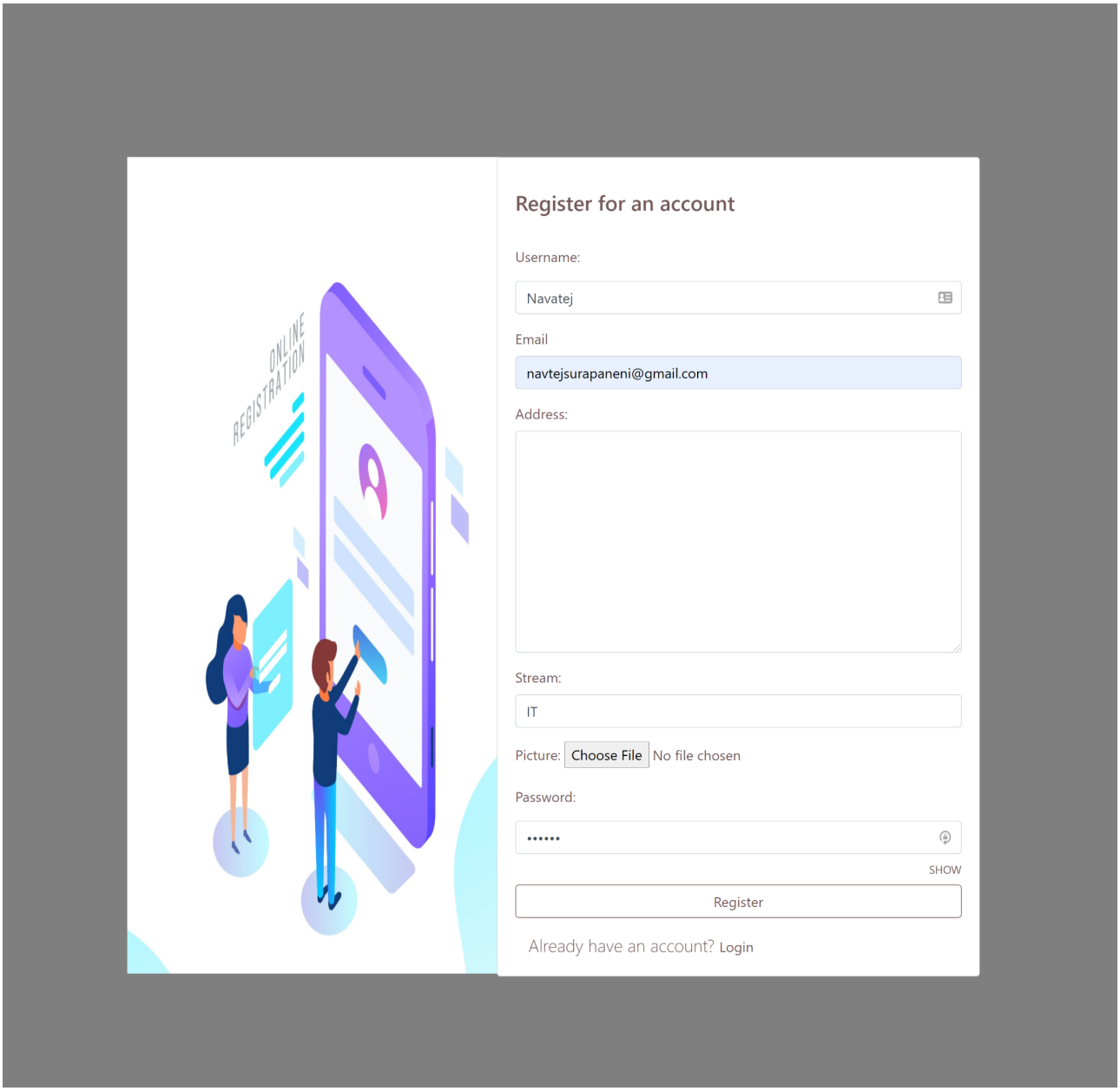


FIG 6.2: User Registration

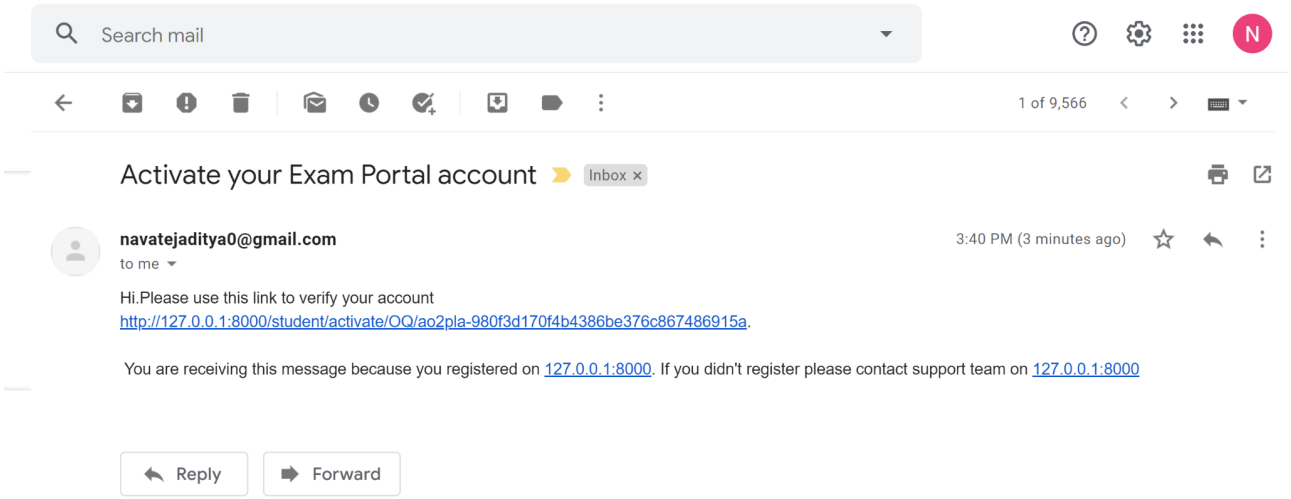


FIG 6.3: Email Verification

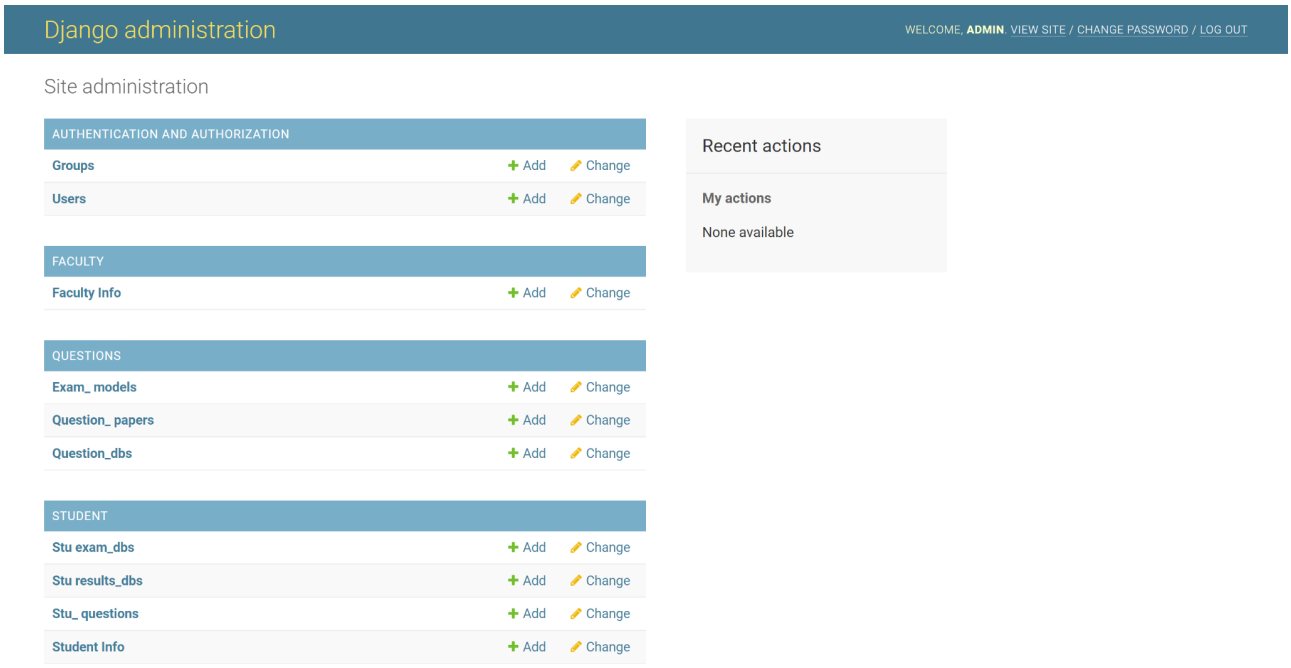


FIG 6.4: Administrator Page

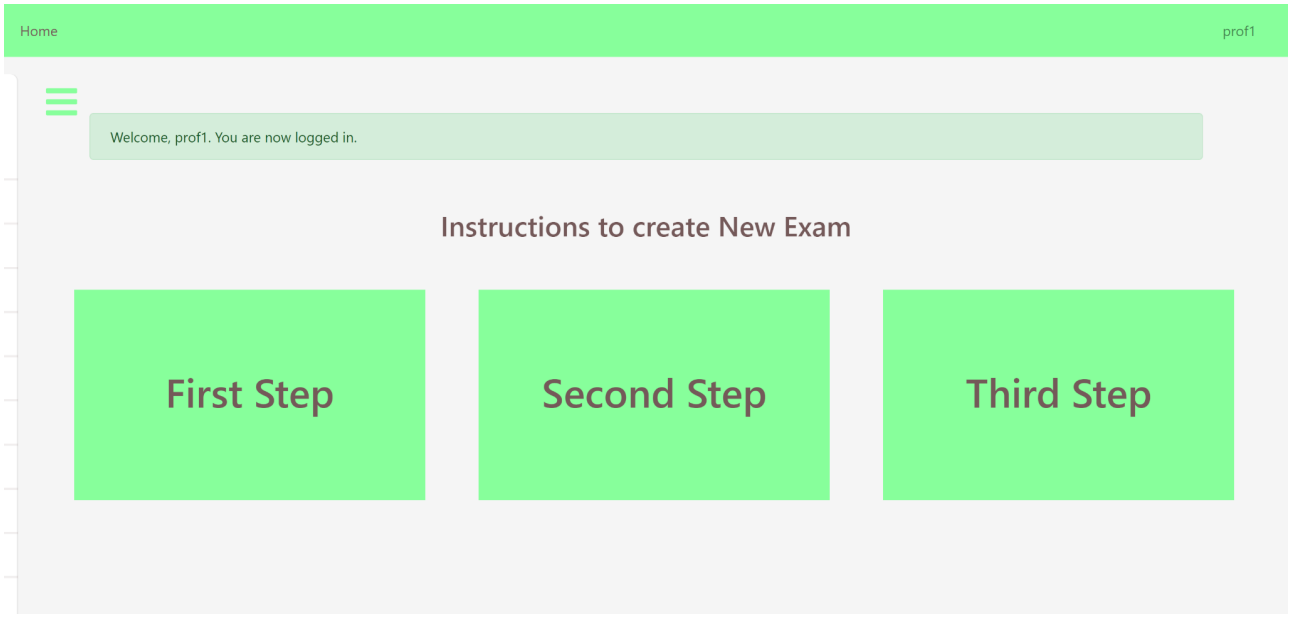


FIG 6.5: Professor Home Page

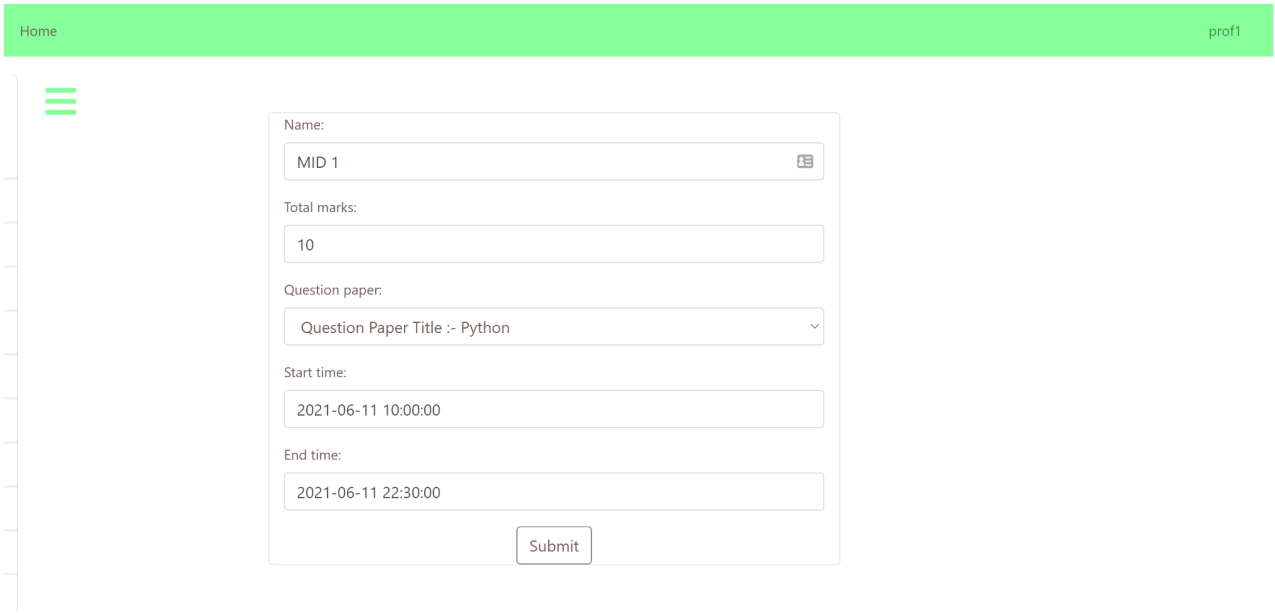


FIG 6.6: Exam Creation Page

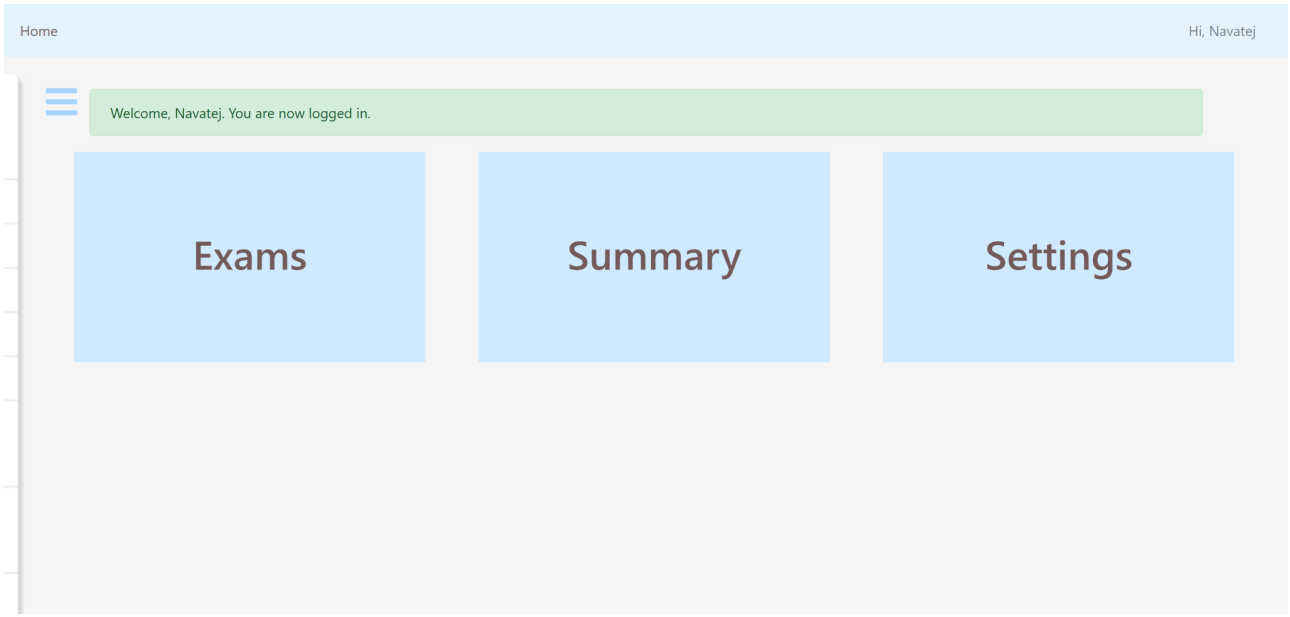


FIG 6.7: Student Home Page

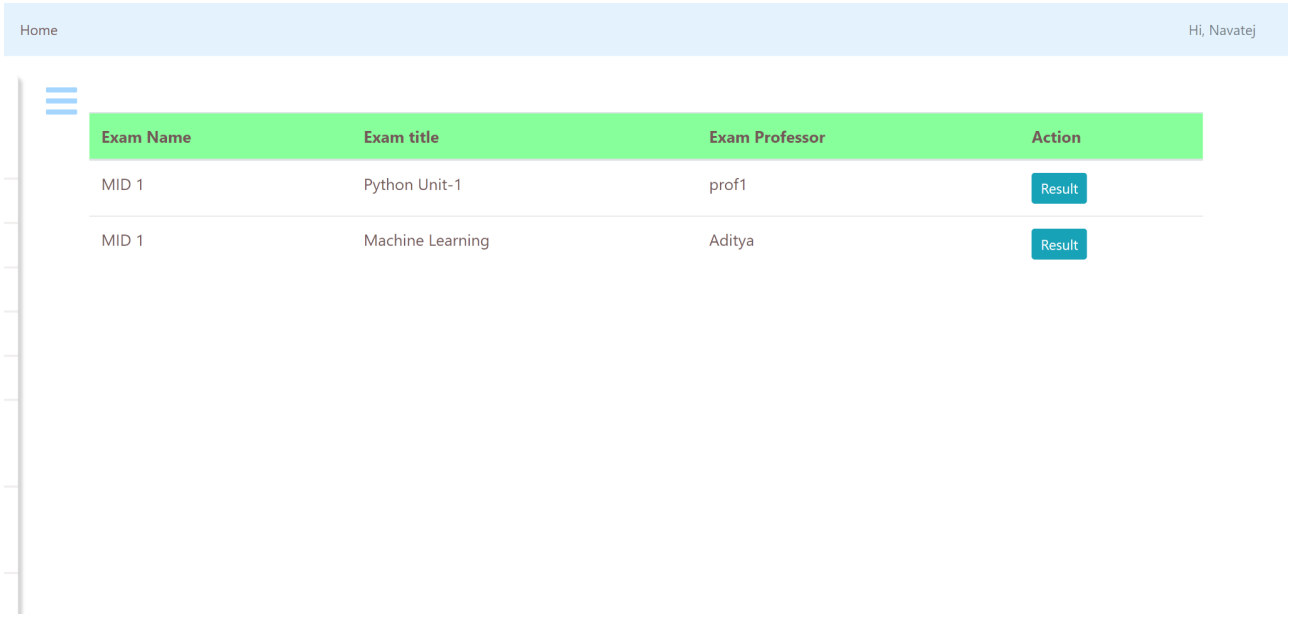


FIG 6.8: Student Exam Result Checking

CHAPTER 7
CONCLUSION

The online examination system is a strong computer network-based application system. This system's design was constructed in Python, and the goal of the system is to investigate a test pattern based on the campus network.

The conventional examination mode's long and complex process has been replaced with this new method. Using the campus network's convenient environment to administer exams to students at any time and from any location considerably simplifies the traditional examination procedure, decreases teachers' workload, and enhances work efficiency and consistency.

Universities and organisations can readily adopt the suggested Online Examination System to make exams more accessible and secure. It will be flexible and easy for further development and maintenance in the future due to its simple architecture.

CHAPTER 8
FUTURE WORK

The proposed system can be made more secure by adding a camera monitoring option, which can be made even more secure with the help of machine learning by automatically alerting the invigilator when the student is not looking at the computer screen.

This system can be modified to automatically create personalized question papers for each student by analyzing their strengths and weaknesses in a particular area of a subject according to their previous grades. With proper datasets, the whole examination process of creating the exams papers and grading can also be done automatically, this will increase the efficiency evenmore by eliminating the need of a faculty member in creating examination papers.

CHAPTER 9

REFERENCES

1. The Research and Design of Online Examination System https://ieeexplore.ieee.org/document/7429241
2. A Study on Web Based Online Examination System https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3611554
3. Li Yueru. Algorithmic Online Examination System Design[J].FuJian computer.2009,1:66-67. https://www.sciencedirect.com/science/article/pii/S1876610212004675
4. Application Research of Web Examination System Based on College https://www.sciencedirect.com/science/article/pii/S1876610212004675
5. Research and Development of Online Examination System https://download.atlantis-press.com/article/4176.pdf
6. A web-based online examination system for computer science education https://www.ijser.org/researchpaper/A-web-based-online-examination-system-for-computer-science-education.pdf
7. Online Exam Software And Online Examination Software https://www.academia.edu/36234682/Online_Exam_Software_And_Online_Examination_Software
8. https://docs.djangoproject.com/en/3.2/
9. https://digilocal.org.uk/wp-content/uploads/2020/03/Quiz-Master.pdf
10. https://pypi.org/project/Django-Verify-Email/
11. https://docs.djangoproject.com/en/3.2/ref/contrib/admin/
12. https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

SOURCE CODE

FACULTY MODULE

FORMS

```
from django.db import models
from django.contrib.auth.models import User


class FacultyInfo(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    address = models.CharField(max_length=200, blank=True)
    subject = models.CharField(max_length=50, blank=True)
    picture = models.ImageField(upload_to = 'faculty_profile_pics', blank=True)

    def __str__(self):
        return self.user.username

    class Meta:
        verbose_name_plural = 'Faculty Info'
```

URLS

```
from django.urls import path
from .views import Register,LoginView,LogoutView
from . import views
from django.contrib.auth import views as auth_views

urlpatterns = [
    path('',views.index,name="faculty-index"),
    path('register/',Register.as_view(),name = "faculty-register"),
    path('login/',LoginView.as_view(),name="faculty-login"),
    path('logout/',LogoutView.as_view(),name="faculty-logout"),
    path('reset-password/',auth_views.PasswordResetView.as_view(template_name="faculty/resetPassword.html"),name="password_reset_faculty"),
    path('reset-password_sent/',auth_views.PasswordResetDoneView.as_view(template_name="faculty/resetPasswordSent.html"),name="password_reset_done_faculty"),
    path('reset/<uidb64>/<token>/',auth_views.PasswordResetConfirmView.as_view(template_name="faculty/setNewPassword.html"),name="password_reset_confirm_faculty"),
    path('reset-password-complete/',auth_views.PasswordResetCompleteView.as_view(template_name="faculty/resetPasswordDone.html"),name="password_reset_complete_faculty"),
]
```

LOGGED IN VIEW

```
from django.shortcuts import render,redirect
from django.views import View
from .forms import FacultyForm,FacultyInfoForm
from .models import FacultyInfo
from django.contrib import auth
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.core.mail import EmailMessage
import threading
from django.contrib.sites.shortcuts import get_current_site
from student.views import EmailThread
from django.contrib.auth.models import User
from django.contrib.auth.models import Group
from questions.views import has_group

@login_required(login_url='faculty-login')
def index(request):
    return render(request,'faculty/index.html')

class Register(View):
    def get(self,request):
        faculty_form = FacultyForm()
        faculty_info_form = FacultyInfoForm()
        return render(request,'faculty/register.html',{'faculty_form':faculty_form,'faculty_info_form':faculty_info_form}

    def post(self,request):
        faculty_form = FacultyForm(data=request.POST)
        faculty_info_form = FacultyInfoForm(data=request.POST)
        email = request.POST['email']

        if faculty_form.is_valid() and faculty_info_form.is_valid():
            faculty = faculty_form.save()
            faculty.set_password(faculty.password)
            faculty.is_active = True
            faculty.is_staff = True
            faculty.save()

            domain = get_current_site(request).domain
            email_subject = 'Activate your Exam Portal Faculty account'
            email_body = "Hi. Please contact the admin team of "+domain+". To register yourself as a professor."+ ".\n\n '
            fromEmail = 'noreply@exam.com'


            email = EmailMessage(
                email_subject,
                email_body,
                fromEmail,
                [email],
            )
            student_info = faculty_info_form.save(commit=False)
            student_info.user = faculty
            if 'picture' in request.FILES:
                student_info.picture = request.FILES['picture']
            student_info.save()
            messages.success(request,"Registered Succesfully. Check Email for confirmation")
            EmailThread(email).start()
            return redirect('faculty-login')
        else:
            print(faculty_form.errors,faculty_info_form.errors)
            return render(request,'faculty/register.html',{'faculty_form':faculty_form,'faculty_info_form':faculty_info_form})

class LoginView(View):
    def get(self,request):
        return render(request,'faculty/login.html')
    def post(self,request):
        username = request.POST['username']
        password = request.POST['password']
        has_grp = False
        if username and password:
            user = auth.authenticate(username=username,password=password)
            exis = User.objects.filter(username=username).exists()
            if exis:
                user_ch = User.objects.get(username=username)
                has_grp = has_group(user_ch,"Professor")
            if user and user.is_active and exis and has_grp:
                auth.login(request,user)
                messages.success(request,"Welcome, "+ user.username + ". You are now logged in.")
                return redirect('faculty-index')
            elif not has_grp and exis:
                messages.error(request,'You dont have permssions to login as faculty. If You think this is a mistake please contact admin')
                return render(request,'faculty/login.html')

            else:
                messages.error(request,'Invalid credentials')
                return render(request,'faculty/login.html')


class LogoutView(View):
    def post(self,request):
        auth.logout(request)
        messages.success(request,'Logged Out')
        return redirect('faculty-login')
```

STUDENT MODULE

EMAIL ALERTS

```
from django.views import View
from django.contrib.auth.models import User
from django.http import JsonResponse
import json
from validate_email import validate_email
from .views import EmailThread
from django.core.mail import EmailMessage

class UsernameValidation(View):
    def post(self,request):
        data = json.loads(request.body)
        username = data['username']

        if not str(username).isalnum():
            return JsonResponse({'username_error':'Username should only contain alphanumeric characters'},status=400)

        if User.objects.filter(username=username).exists():
            return JsonResponse({'username_error':'Username Exists'},status=409)

        return JsonResponse({'username_valid':True})

class EmailValidationView(View):
    def post(self,request):
        data = json.loads(request.body)
        email = data['email']

        if not validate_email(email):
            return JsonResponse({'email_error':'Email is invalid'},status = 400)

        if User.objects.filter(email = email ).exists():
            return JsonResponse({'email_error':'email exists'},status = 409)

        return JsonResponse({'email_valid':True})


  class Cheating(View):
      def get(self,request,professorname):
          student = str(request.user.username)
          email = User.objects.get(username=professorname).email
          email_subject = 'Student Cheating'
          email_body = 'Student caught changing window for 5 times. Student username is :' + student
          fromEmail = 'noreply@exam.com'
          email_obj = EmailMessage(
              email_subject,
              email_body,
              fromEmail,
              [email],
          )
          EmailThread(email_obj).start()
          return JsonResponse({'sent':True})
```

URLs

```
from django.urls import path
from . import views
from .views import Register,LoginView,LogoutView,VerificationView
from .api import UsernameValidation,EmailValidationView,Cheating
from django.views.decorators.csrf import csrf_exempt
from django.contrib.auth import views as auth_views

urlpatterns = [
    path('',views.index,name = "index"),
    path('register/',Register.as_view(),name="register"),
    path('login/',LoginView.as_view(),name="login"),
    path('logout/',LogoutView.as_view(),name="logout"),
    path('username-validate',UsernameValidation.as_view(),name="username-validate"),
    path('cheat/<str:professorname>',Cheating.as_view(),name="cheat"),
    path('email-validate',EmailValidationView.as_view(),name="email-validate"),
    path('activate/<uidb64>/<token>',VerificationView.as_view(),name = 'activate'),
    path('reset-password/',auth_views.PasswordResetView.as_view(template_name="student/resetPassword.html"),name="password_reset"),
    path('reset-password_sent/',auth_views.PasswordResetDoneView.as_view(template_name="student/resetPasswordSent.html"),name="password_reset_done"),
    path('reset/<uidb64>/<token>/',auth_views.PasswordResetConfirmView.as_view(template_name="student/setNewPassword.html"),name="password_reset_confirm"),
    path('reset-password-complete/',auth_views.PasswordResetCompleteView.as_view(template_name="student/resetPasswordDone.html"),name="password_reset_complete"),
]
```

EXAMINATION

```
from django.db import models
from django.contrib.auth.models import User
from questions.question_models import Question_DB
from questions.questionpaper_models import Question_Paper

class StudentInfo(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    address = models.CharField(max_length=200, blank=True)
    stream = models.CharField(max_length=50, blank=True)
    picture = models.ImageField(upload_to = 'student_profile_pics', blank=True)

    def __str__(self):
        return self.user.username

    class Meta:
        verbose_name_plural = 'Student Info'

class Stu_Question(Question_DB):
    professor = None
    student = models.ForeignKey(User, limit_choices_to={'groups__name': "Student"}, on_delete=models.CASCADE, null=True)
    choice = models.CharField(max_length=3, default="E")

    def __str__(self):
        return str(self.student.username) + " "+ str(self.qno) +"-Stu_QuestionDB"


class StuExam_DB(models.Model):
    student = models.ForeignKey(User, limit_choices_to={'groups__name': "Student"}, on_delete=models.CASCADE, null=True)
    examname = models.CharField(max_length=100)
    qpaper = models.ForeignKey(Question_Paper, on_delete=models.CASCADE, null=True)
    questions = models.ManyToManyField(Stu_Question)
    score = models.IntegerField(default=0)
    completed = models.IntegerField(default=0)

    def __str__(self):
        return str(self.student.username) +" " + str(self.examname) + " " + str(self.qpaper.qPaperTitle) + "-StuExam_DB"


class StuResults_DB(models.Model):
    student = models.ForeignKey(User, limit_choices_to={'groups__name': "Student"}, on_delete=models.CASCADE, null=True)
    exams = models.ManyToManyField(StuExam_DB)

    def __str__(self):
        return str(self.student.username) +" -StuResults_DB"
```

STUDENT VIEW

```
from django.shortcuts import render,redirect
from django.views import View
from .forms import StudentForm, StudentInfoForm
from django.contrib.auth.decorators import login_required
from django.contrib import auth
from django.contrib import messages
from django.contrib.sites.shortcuts import get_current_site
from django.utils.http import urlsafe_base64_decode, urlsafe_base64_encode
from django.urls import reverse
from django.utils.encoding import force_bytes, force_text, DjangoUnicodeDecodeError
from .utils import account_activation_token
from django.core.mail import EmailMessage
import threading
from django.contrib.auth.models import User
from studentPreferences.models import StudentPreferenceModel
from django.contrib.auth.models import Group

@login_required(login_url='login')
def index(request):
    return render(request,'student/index.html')

class Register(View):
    def get(self,request):
        student_form = StudentForm()
        student_info_form = StudentInfoForm()
        return render(request,'student/register.html',{'student_form':student_form,'student_info_form':student_info_form})

    def post(self,request):
        student_form = StudentForm(data=request.POST)
        student_info_form = StudentInfoForm(data=request.POST)
        email = request.POST['email']

        if student_form.is_valid() and student_info_form.is_valid():
            student = student_form.save()
            student.set_password(student.password)
            student.is_active = False
            my_group = Group.objects.get_or_create(name='Student')
            my_group[0].user_set.add(student)
            student.save()

            uidb64 = urlsafe_base64_encode(force_bytes(student.pk))
            domain = get_current_site(request).domain
            link = reverse('activate',kwargs={'uidb64':uidb64,'token':account_activation_token.make_token(student)})
            activate_url = 'http://' + domain +link
            print(activate_url)
            email_subject = 'Activate your Exam Portal account'
            email_body = 'Hi.Please use this link to verify your account\n' + activate_url + ".\n\n You are receiving this message because you registered on
            fromEmail = 'noreply@exam.com'
            email = EmailMessage(
                email_subject,
                email_body,
                fromEmail,
                [email],
            )
            student_info = student_info_form.save(commit=False)
            student_info.user = student
            if 'picture' in request.FILES:
                student_info.picture = request.FILES['picture']
            student_info.save()
            messages.success(request,"Registered Succesfully. Check Email for confirmation")
            EmailThread(email).start()
            return redirect('login')
        else:
            print(student_form.errors,student_info_form.errors)
            return render(request,'student/register.html',{'student_form':student_form,'student_info_form':student_info_form})

class LoginView(View):
    def get(self,request):
        return render(request,'student/login.html')
    def post(self,request):
        username = request.POST['username']
        password = request.POST['password']

        if username and password:
            exis = User.objects.filter(username=username).exists()
            if exis:
                user_ch = User.objects.get(username=username)
                if user_ch.is_staff:
                    messages.error(request,"You are trying to login as student, but you have registered as faculty. We are redirecting you to faculty login.
                    return redirect('faculty-login')
            user = auth.authenticate(username=username,password=password)
```

```
            if user:
                if user.is_active:
                    auth.login(request,user)
                    student_pref = StudentPreferenceModel.objects.filter(user = request.user).exists()
                    email = User.objects.get(username=username).email

                    email_subject = 'You Logged into your Portal account'
                    email_body = "If you think someone else logged in. Please contact support or reset your password.\n\n
                    fromEmail = 'noreply@exam.com'
                    email = EmailMessage(
                        email_subject,
                        email_body,
                        fromEmail,
                        [email],
                    )
                    if student_pref :
                        student = StudentPreferenceModel.objects.get(user=request.user)
                        sendEmail = student.sendEmailOnLogin
                    if not student_pref :
                        EmailThread(email).start()
                    elif sendEmail:
                        EmailThread(email).start()
                    messages.success(request,"Welcome, "+ user.username + ". You are now logged in.")

                    return redirect('index')

                else:
                    user_n = User.objects.filter(username=username).exists()
                    if user_n:
                        user_v = User.objects.get(username=username)
                        if user_v.is_active:
                            messages.error(request,'Invalid credentials')
                            return render(request,'student/login.html')
                        else:
                            messages.error(request,'Account not Activated')
                            return render(request,'student/login.html')

            messages.error(request,'Please fill all fields')
            return render(request,'student/login.html')


            messages.error(request,'Please fill all fields')
            return render(request,'student/login.html')

    class LogoutView(View):
        def post(self,request):
            auth.logout(request)
            messages.success(request,'Logged Out')
            return redirect('login')

    class EmailThread(threading.Thread):
        def __init__(self,email):
            self.email = email
            threading.Thread.__init__(self)


        def run(self):
            self.email.send(fail_silently = False)

    class VerificationView(View):
        def get(self,request,uidb64,token):
            try:
                id = force_text(urlsafe_base64_decode(uidb64))
                user = User.objects.get(pk=id)
                if not account_activation_token.check_token(user,token):
                    messages.error(request,"User already Activated. Please Proceed With Login")
                    return redirect("login")
                if user.is_active:
                    return redirect('login')
                user.is_active = True
                user.save()
                messages.success(request,'Account activated Sucessfully')
                return redirect('login')
            except Exception as e:
                raise e
            return redirect('login')
```

EXAMINATION MODULE

CREATING QUESTIONS

```
from django.db import models
from django.forms import ModelForm
from django.contrib.auth.models import User
from django import forms

class Question_DB(models.Model):
    professor = models.ForeignKey(User, limit_choices_to={'groups__name': "Professor"}, on_delete=models.CASCADE, null=True)
    qno = models.AutoField(primary_key=True)
    question = models.CharField(max_length=100)
    optionA = models.CharField(max_length=100)
    optionB = models.CharField(max_length=100)
    optionC = models.CharField(max_length=100)
    optionD = models.CharField(max_length=100)
    answer = models.CharField(max_length=200)
    max_marks = models.IntegerField(default=0)

    def __str__(self):
        return f'Question No.{self.qno}: {self.question} \t\t Options: \nA. {self.optionA} \nB.{self.optionB} \nC.{self.optionC} \nD.{self.optionD} '


class QForm(ModelForm):
    class Meta:
        model = Question_DB
        fields = '__all__'
        exclude = ['qno', 'professor']
        widgets = {
            'question': forms.TextInput(attrs = {'class':'form-control'}),
            'optionA': forms.TextInput(attrs = {'class':'form-control'}),
            'optionB': forms.TextInput(attrs = {'class':'form-control'}),
            'optionC': forms.TextInput(attrs = {'class':'form-control'}),
            'optionD': forms.TextInput(attrs = {'class':'form-control'}),
            'answer': forms.TextInput(attrs = {'class':'form-control'}),
            'max_marks': forms.NumberInput(attrs = {'class':'form-control'}),
        }
```

QUESTION PAPERS

```
from django.db import models
from django.forms import ModelForm
from django.contrib.auth.models import User
from .question_models import Question_DB
from django import forms

class Question_Paper(models.Model):
    professor = models.ForeignKey(User, limit_choices_to={'groups__name': "Professor"}, on_delete=models.CASCADE)
    qPaperTitle = models.CharField(max_length=100)
    questions = models.ManyToManyField(Question_DB)

    def __str__(self):
        return f' Question Paper Title :- {self.qPaperTitle}\n'


class QPForm(ModelForm):
    def __init__(self,professor,*args,**kwargs):
        super (QPForm,self ).__init__(*args,**kwargs)
        self.fields['questions'].queryset = Question_DB.objects.filter(professor=professor)

    class Meta:
        model = Question_Paper
        fields = '__all__'
        exclude = ['professor']
        widgets = {
            'qPaperTitle': forms.TextInput(attrs = {'class':'form-control'})
        }
```

LINKING

```
from django.db import models
from django.forms import ModelForm
from django.contrib.auth.models import User
from datetime import datetime
from .questionpaper_models import Question_Paper
from django import forms

class Exam_Model(models.Model):
    professor = models.ForeignKey(User, limit_choices_to={'groups__name': "Professor"}, on_delete=models.CASCADE)
    name = models.CharField(max_length=50)
    total_marks = models.IntegerField()
    question_paper = models.ForeignKey(Question_Paper, on_delete=models.CASCADE, related_name='exams')
    start_time = models.DateTimeField(default=datetime.now())
    end_time = models.DateTimeField(default=datetime.now())

    def __str__(self):
        return self.name


class ExamForm(ModelForm):
    def __init__(self,professor,*args,**kwargs):
        super (ExamForm,self ).__init__(*args,**kwargs)
        self.fields['question_paper'].queryset = Question_Paper.objects.filter(professor=professor)

    class Meta:
        model = Exam_Model
        fields = '__all__'
        exclude = ['professor']
        widgets = {
            'name': forms.TextInput(attrs = {'class':'form-control'}),
            'total_marks' : forms.NumberInput(attrs = {'class':'form-control'}),
            'start_time': forms.DateTimeInput(attrs = {'class':'form-control'}),
            'end_time': forms.DateTimeInput(attrs = {'class':'form-control'})
        }
```

URLs

```
from django.urls import path
from . import views
urlpatterns = [
    path('prof/viewexams/',views.view_exams_prof,name="view_exams"),
    path('prof/viewpreviousexams/',views.view_previousexams_prof,name="faculty-previous"),
    path('prof/viewresults/',views.view_results_prof,name="faculty-result"),
    path('prof/addquestions/',views.add_questions,name="faculty-addquestions"),
    path('prof/addnewquestionpaper/',views.add_question_paper,name="faculty-add_question_paper"),
    path('prof/viewstudents/',views.view_students_prof,name="faculty-student"),
    path('student/viewexams/',views.view_exams_student,name="view_exams_student"),
    path('student/previous/',views.student_view_previous,name="student-previous"),
    path('student/appear/<int:id>',views.appear_exam,name = "appear-exam"),
    path('student/result/<int:id>',views.result,name = "result"),
    path('student/attendance/',views.view_students_attendance,name="view_students_attendance")
]
```

MODELLING THE EXAM

```
from django.shortcuts import render,redirect
from django.contrib.auth.models import User
from .models import *
from django.contrib.auth.models import Group
from student.models import *
from django.utils import timezone
from student.models import StuExam_DB,StuResults_DB
from questions.questionpaper_models import QPForm
from questions.question_models import QForm
from django.utils import timezone
from django.contrib.auth.decorators import login_required

def has_group(user, group_name):
    group = Group.objects.get(name=group_name)
    return True if group in user.groups.all() else False

@login_required(login_url='faculty-login')
def view_exams_prof(request):
    prof = request.user
    prof_user = User.objects.get(username=prof)
    permissions = False
    if prof:
        permissions = has_group(prof,"Professor")
    if permissions:
        new_Form = ExamForm(prof_user)
        if request.method == 'POST' and permissions:
            form = ExamForm(prof_user,request.POST)
            if form.is_valid():
                exam = form.save(commit=False)
                exam.professor = prof
                exam.save()
                form.save_m2m()
                return redirect('view_exams')

        exams = Exam_Model.objects.filter(professor=prof)
        return render(request, 'exam/mainexam.html', {
            'exams': exams, 'examform': new_Form, 'prof': prof,
        })
    else:
        return redirect('view_exams_student')
```

```python
@login_required(login_url='faculty-login')
def add_question_paper(request):
    prof = request.user
    prof_user = User.objects.get(username=prof)
    permissions = False
    if prof:
        permissions = has_group(prof,"Professor")
    if permissions:
        new_Form = QPForm(prof_user)
        if request.method == 'POST' and permissions:
            form = QPForm(prof_user,request.POST)
            if form.is_valid():
                exam = form.save(commit=False)
                exam.professor = prof_user
                exam.save()
                form.save_m2m()
                return redirect('faculty-add_question_paper')

        exams = Exam_Model.objects.filter(professor=prof)
        return render(request, 'exam/addquestionpaper.html', {
            'exams': exams, 'examform': new_Form, 'prof': prof,
        })
    else:
        return redirect('view_exams_student')

@login_required(login_url='faculty-login')
def add_questions(request):
    prof = request.user
    prof_user = User.objects.get(username=prof)
    permissions = False
    if prof:
        permissions = has_group(prof,"Professor")
    if permissions:
        new_Form = QForm()
        if request.method == 'POST' and permissions:
            form = QForm(request.POST)
            if form.is_valid():
                exam = form.save(commit=False)
                exam.professor = prof_user
                exam.save()
                form.save_m2m()
                return redirect('faculty-addquestions')
```

```
            exams = Exam_Model.objects.filter(professor=prof)
            return render(request, 'exam/addquestions.html', {
                'exams': exams, 'examform': new_Form, 'prof': prof,
            })
        else:
            return redirect('view_exams_student')

@login_required(login_url='faculty-login')
def view_previousexams_prof(request):
    prof = request.user
    student = 0
    exams = Exam_Model.objects.filter(professor=prof)
    return render(request, 'exam/previousexam.html', {
        'exams': exams,'prof': prof
    })

@login_required(login_url='login')
def student_view_previous(request):
    exams = Exam_Model.objects.all()
    list_of_completed = []
    list_un = []
    for exam in exams:
        if StuExam_DB.objects.filter(examname=exam.name ,student=request.user).exists():
            if StuExam_DB.objects.get(examname=exam.name,student=request.user).completed == 1:
                list_of_completed.append(exam)
        else:
            list_un.append(exam)

    return render(request,'exam/previousstudent.html',{
        'exams':list_un,
        'completed':list_of_completed
    })



@login_required(login_url='faculty-login')
def view_students_prof(request):
    students = User.objects.filter(groups__name = "Student")
    student_name = []
    student_completed = []
    count = 0
    dicts = {}
    examn = Exam_Model.objects.filter(professor=request.user)
    for student in students:
        student_name.append(student.username)
        count = 0
        for exam in examn:
            if StuExam_DB.objects.filter(student=student,examname=exam.name,completed=1).exists():
                count += 1
            else:
                count += 0
        student_completed.append(count)
    i = 0
    for x in student_name:
        dicts[x] = student_completed[i]
        i+=1
    return render(request, 'exam/viewstudents.html', {
        'students':dicts
    })

@login_required(login_url='faculty-login')
def view_results_prof(request):
    students = User.objects.filter(groups__name = "Student")
    dicts = {}
    prof = request.user
    professor = User.objects.get(username=prof.username)
    examn = Exam_Model.objects.filter(professor=professor)
    for exam in examn:
        if StuExam_DB.objects.filter(examname=exam.name,completed=1).exists():
            students_filter = StuExam_DB.objects.filter(examname=exam.name,completed=1)
            for student in students_filter:
                key = str(student.student) + " " + str(student.examname) + " " + str(student.qpaper.qPaperTitle)
                dicts[key] = student.score
    return render(request, 'exam/resultsstudent.html', {
        'students':dicts
    })
```

```python
@login_required(login_url='login')
def view_exams_student(request):
    exams = Exam_Model.objects.all()
    list_of_completed = []
    list_un = []
    for exam in exams:
        if StuExam_DB.objects.filter(examname=exam.name ,student=request.user).exists():
            if StuExam_DB.objects.get(examname=exam.name,student=request.user).completed == 1:
                list_of_completed.append(exam)
        else:
            list_un.append(exam)

    return render(request,'exam/mainexamstudent.html',{
        'exams':list_un,
        'completed':list_of_completed
    })

@login_required(login_url='login')
def view_students_attendance(request):
    exams = Exam_Model.objects.all()
    list_of_completed = []
    list_un = []
    for exam in exams:
        if StuExam_DB.objects.filter(examname=exam.name ,student=request.user).exists():
            if StuExam_DB.objects.get(examname=exam.name,student=request.user).completed == 1:
                list_of_completed.append(exam)
        else:
            list_un.append(exam)

    return render(request,'exam/attendance.html',{
        'exams':list_un,
        'completed':list_of_completed
    })

def convert(seconds):
    min, sec = divmod(seconds, 60)
    hour, min = divmod(min, 60)
    min += hour*60
    return "%02d:%02d" % (min, sec)
```

```python
@login_required(login_url='login')
def appear_exam(request,id):
    student = request.user
    if request.method == 'GET':
        exam = Exam_Model.objects.get(pk=id)
        time_delta = exam.end_time - exam.start_time
        time = convert(time_delta.seconds)
        time = time.split(":")
        mins = time[0]
        secs = time[1]
        context = {
            "exam":exam,
            "question_list":exam.question_paper.questions.all(),
            "secs":secs,
            "mins":mins
        }
        return render(request,'exam/giveExam.html',context)
    if request.method == 'POST' :
        student = User.objects.get(username=request.user.username)
        paper = request.POST['paper']
        examMain = Exam_Model.objects.get(name = paper)
        stuExam = StuExam_DB.objects.get_or_create(examname=paper, student=student,qpaper = examMain.question_paper)[0]

        qPaper = examMain.question_paper
        stuExam.qpaper = qPaper

        qPaperQuestionsList = examMain.question_paper.questions.all()
        for ques in qPaperQuestionsList:
            student_question = Stu_Question(student=student,question=ques.question, optionA=ques.optionA, optionB=ques.optionB,optionC=ques.optionC, optionD=ques.optionD,
            answer=ques.answer)
            student_question.save()
            stuExam.questions.add(student_question)
            stuExam.save()

        stuExam.completed = 1
        stuExam.save()
        examQuestionsList = StuExam_DB.objects.filter(student=request.user,examname=paper,qpaper=examMain.question_paper,questions__student = request.user)[0]
        #examQuestionsList = stuExam.questions.all()
        examScore = 0
        list_i = examMain.question_paper.questions.all()
        queslist = examQuestionsList.questions.all()
        i = 0


        for j in range(list_i.count()):
            ques = queslist[j]
            max_m = list_i[i].max_marks
            ans = request.POST.get(ques.question, False)
            if not ans:
                ans = "E"
            ques.choice = ans
            ques.save()
            if ans.lower() == ques.answer.lower() or ans == ques.answer:
                examScore = examScore + max_m
            i+=1

        stuExam.score = examScore
        stuExam.save()
        stu = StuExam_DB.objects.filter(student=request.user,examname=examMain.name)
        results = StuResults_DB.objects.get_or_create(student=request.user)[0]
        results.exams.add(stu[0])
        results.save()
        return redirect('view_exams_student')

@login_required(login_url='login')
def result(request,id):
    student = request.user
    exam = Exam_Model.objects.get(pk=id)
    score = StuExam_DB.objects.get(student=student,examname=exam.name,qpaper=exam.question_paper).score
    return render(request,'exam/result.html',{'exam':exam,"score":score})
```