**AMAR SINGH COLLEGE**

**Gogji Bagh, Srinagar**



Project Report on

**“ONLINE EXAMINATION SYSTEM”**

For Partial Fulfilment Of

**Bachelor’s of Computer Applications**

Submitted by:

Mohammad Owais Mir (Roll No: 1909740 / 5012)

Jahangir Ahmad Lone (Roll No: 1909748 / 5025)

Adnan Farooq (Roll No: 1909775 / 5032)

Guided by:

**Professor Arshad Hussain**

Submitted to:

Department of Computer Science

For Academic Year

2022

**AMAR SINGH COLLEGE**

**GOGJI BAGH, SRINAGAR-190008**

**DEPARTMENT OF COMPUTER APPLICATIONS,**

****

**CERTIFICATE**

We **Mohammad Owais Mir** (1909740)**, Jahangir Ahmad Lone** (1909748), **Adnan Farooq (**1909775**),** hereby declare that the work, which is being presented in the project entitled **”ONLINE EXAMINATION SYSTEM”** for the partial fulfillment of the award of BACHELOR’S OF COMPUTER APPLICATIONS(BCA) degree in the year 2022 is an authentic record of our own work carried out under the supervision of **PROFESSOR ARSHAD HUSSAIN**, Department of Computer Application, Amar Singh College Srinagar. The matter embodied in this project has not been submitted by us to any other degree.

Date: \_\_\_\_\_\_\_\_\_\_\_\_

1. Mohammad Owais Mir 2. Jahangir Ahmad Lone 3. Adnan Farooq

This is to certify that the above statements made by the candidates are correct to the best of my knowledge.

**Supervisor Head of Department**

**Prof. Arshad Hussain Mr. Arshad Ahmad Yatoo**

**ACKNOWLEDGEMENT**

We would like thank to all those who are involved in this endeavor for their kind cooperation for its successful completion. At the outset, we wish to express our sincere gratitude to all those people who have helped us to complete this project in an efficient manner.

We offer our special thanks to our Project guide**Professor Arshad Hussain (Assistant Professor)** DEPARTMENT OF COMPUTER APPLICATIONS, AMAR SINGH COLLEGE SRINAGAR, without whose help and support throughout this project would not have been this success. We would like to thank **Prof. Arshad Ahmad Yatoo**, HEAD DEPARTMENT OF COMPUTER APPLICATIONS, who gave opportunity to do this project at an extreme organization.

Last but not least we are thankful to all faculty members and lab instructors without whose support at various stages, this project would not have materialized.

**Mohammad Owais**

**Jahangir Ahmad**

**Adnan Farooq**

**INDEX**

1. **Introduction** …...……………………………………………**11-15**

1.1 Abstract………..………………………………….……….11

* 1. Overview…………………………………..…….................12
  2. Existing system …………...………………...…….………13

1.3.1 Problems in existing System…………………..........13

* 1. Proposed system……………………..……….….……...14
     1. Overview…………………………..……………….14

1.4.2 Features of proposed system. ………..…………….14

* 1. Scope. …………………………………..……………..15

1. **Agile Approach:** …………………………………………….**16-23**
   1. Methodology. …………………………...…….………..…17
   2. Agile approach.………….……………………………...**18-19**
   3. Analysis and design. ……………….………….…..............20
      1. Analysis phase. …………………………....20

2.3.2 Preliminary investigation. ……………........21

* 1. Feasibility study…………………………….……………...21
     1. Economic feasibility. ………….……..........22
     2. Technical feasibility. ………………............22

2.4.3 Operational feasibility...……..…………......22

2.5 Design Phase………………………………..…...................23

**3.** **ER Diagram**…..……………….………………..................**24-26**

3.1 Elements of ER…………….…………………………….25

3.2 ER Diagram Components…..……..……………………..25

3.3 ER Diagram…….……………….……............................26

**4**. **Data Flow Diagrams**……..………………….………........**27-29**

4.1 0- level DFD………..…….…...….……….................29

4.2 1-level DFD…………...……….…....…….................29

**5. Use Case Models**…………………….…...……...……..…**31-35**

5.1 Use Case Diagram of Admin…..………………….....32

* 1. Use Case Diagram of Student…….…...……………33

5.3 Use Case Diagram of HOD………….......................34

5.4 Use Case Diagram of Faculty………………………35

**6. Activity Diagram**:.……..…………………...………..........**36-41**

* 1. Activity diagram for Login……...…………………..37
  2. Activity diagram for Admin………...…….…………38

6.3 Activity diagram for HOD.………………………….39

6.4 Activity diagram for Faculty.……………………….40

6.5 Activity diagram for Student…….………………….41

**7. Overview of Technologies**. ……………………….........**42-57**

7.1 Overview……………………………………………….43

* + 1. Front End………………....……..................43
    2. Architecture……………………………......43
    3. Back End………………………………......43
    4. Other Frame Works…………………….....43

**7.2** System Specification .…...…………………….……44

7.2.1 Hardware Requirements ..................................44

7.2.2 Processor……………………………………44

7.2.3 RAM…………………………………….......44

7.2.4 Hard Disk…………………………………...44

7.2.5 Cache………………………………………..44

* + 1. Software Requirements……………………..44

7.2.7 Man Power Requirements……..……………44

**7.3** Language Specification. ……………..……………...45

7.3.1. C#.................................................................45

* + 1. Characteristics of C#....................................45

7.4 .NET Core. ……………………………………………46

7.4.1 Common language runtime ………………47

7.4.2 .Net Core library………...............................47

7.4.3 Design Features of .Net Core……………..48

* + 1. .Net Core Working…….………………….48

7.4.5 ASP .NET Core MVC…………………….49

* 1. Advantages of C# and .NET Programming.……....…...51
  2. SQL Server…………….……………………………….52

7.6.1 Features of SQL Server 2019…………….52

7.6.2 Advantages of SQL server 2019………...53

* 1. Entity Core 6.x…………………………...…………….54
  2. LINQ(C#)……………. ……………………..................54
  3. HTML…………….... …………………………............54

7.9.1 Applications of HTML……………………55

* 1. CSS……………………...……………………………...55
     1. Features of CSS. ………………...……….56

7.10.2 CSS Applications………………………...56

* 1. Java script………………………………………….....57
  2. Bootstrap…………………………………………......57

**8. Database Design:** …………….……………..……….......**58-64**

* 1. Users Table……………………………………….......59
  2. Department Table………………………………..…...60
  3. Course Table………………………………………….60
  4. Semester Table………………………………………..60

8.5 Student Table……………………………………........61

* 1. Subject Table…………………………………………61
  2. Question Bank Table…………………………………62
  3. Questions Table….…………………………………...62
  4. Paper Details……………………………….…...........63
  5. Results Table………………………….………….......63
  6. Score Table…………………………………………...64
  7. Final Results………………………………………….64
  8. Toppers Table…………………………………………64

**9. Testing Techniques** ……………..…………………..**65-71**

9.1 Design of test cases. ……………………………………..66

9.1.1 Black box testing …………………………....66

* + 1. White box testing…………………………….67
  1. Unit testing.………………………….………………..68
  2. Integration testing. …………………………………....68
  3. System Testing……………………...………………...68
  4. Acceptance Testing…………………………….……..69
  5. Figure Testing process…………………………….…..70
  6. Test cases. ………………………………………….…71

**10. Implementation………………………………………….73-117**

* 1. Domain Models. ………………………………………74
     1. Base Model………….…………………...74
     2. User Model. ……………………………...74
     3. Course Model. ………………...................75
     4. Department Model……………………….75
     5. Employee Model…………………………75
     6. Paper Model……………………………...76
  2. Repository Pattern……………………………………..77
     1. IRepository……………………………….77
     2. Repository………………………….....78-80
     3. Account Repository…………………..81-83
  3. Dbcontext……………………………………………...84
  4. Account Manager…………………………...……85-98
  5. Web App Controllers………………………………...99
     1. Account Controller…………………99-109
     2. Paper Controller……………………110-111
     3. Question Controller………………….....112
  6. Design……………………………………………......113
     1. Layout……………………………..113-114
     2. Enter Exam………………………..115-117

**11. Future scope.** …………………………………….......**118-119**

**12. Conclusion.** …………………………………………..120-121

**13. Screenshots:** …………………………………………..**122-129**

* 1. Home Page. ………………………………………..123
  2. Login Page. ………………………………………124
  3. Add Department...………………………………….125
  4. Add Student……...…………………………............126
  5. Question Bank…………..………………………….127
  6. Add Paper……...…………………………………...128
  7. Exam…...……………………………………..........129

**14. Bibliography:** ………………………………………...**130-132**

* 1. Books. ……………………………………................131
  2. Websites. …………………………………………....132

Chapter 1

**INTRODUCTION**

* 1. **Abstract**

Online Examination System is a web based system that is able to manage the student exam by computerized way. Give fast service to the students and the exam related staffs such as department, faculty and head of the department. It is a web based application that will conduct exams online at any institute level.

To identify the existing problems and to develop online examination we follow the software lifecycle. These are: feasibility study, requirement analysis, system design, implementation and testing. This lifecycle helps to identify the problems faced in existing system and give the guidelines and required information to develop the proposed system.

There are many technologies involved to build the Online Examination System, one of them is ASP.NET MVC Core. ASP.NET is a web platform that provides all the services that you require to build enterprise applications, client server architecture. ASP.NET MVC core is built on open protocols. Applications can be written in any language that is compatible with the common language runtime (CLR), including Visual Basic and C# etc.

* 1. **Overview**

Present age is the age of technology and almost every manual work done before has been overpowered by the new technology. Even technology is replacing itself day by day. So, in this present era, we chose to contribute a little by developing an Online Examination System for conducting exams online and get rid of the traditional one. Online examination has much advantages over the offline examination. It has now become fast growing method because of its accuracy, efficiency and speed as compared to the existing one.

Online Examination System is a web application created for conducting the exams online at any institutional level. This is a simplest to use web-based application almost for all educational institutions to conduct exams online. The offline examination has its drawbacks so to make the examination easy and efficient and to get rid of the existing system’s drawbacks this web app is going to be helpful and useful. It can be accessed throughout an institution with proper authentication and authorization.

We have developed an Online Examination System with Timer using ASP .NET MVC Core technology. The system has four types of users which are the Admin, the HOD, the faculty and the Examinee/User/Student. The Admin will be using the admin panel of the system, the HOD and the faculty are in charge of populating and maintaining the system’s data. The Admin will manage the list of courses, exams, exam questionnaires and examinee. He/ She can view the exam results and feedbacks of the examinees. The users can only give the exam by logging in with their credentials, view their results and send feedback to the admin.

## **EXISTING SYSTEM**

## 

 All processes in existing system are handled manually. All the work that is done in the existing system is done by the human intervention.

As all the work is done manually, there were a lot of workload on invigilator and it also increases the maximum chances of errors. This is so slow and time consuming. Due to increase in number of students the watch and checking process become more difficult.

1.3.1 Problems faced in existing system are as follows-

* The current system is very time consuming.
* More space and more paper is required.
* More people are required for more students.
* The existing system is more error prone as the whole work is done manually.
* It is not space compatible.

## **PROPOSED SYSTEM**

* + 1. Overview
* Online examination system has become fast growing method now because of its accuracy, efficiency and speed as compared to the traditional one.
* It is a technology-driven way to simplify examination activities like defining exam patterns with question banks, defining exam timer, objective/ subjective question sections, conducting exams using the computer or mobile devices in a paperless manner.
* Online Examination System is a cost-effective, scalable way to convert traditional pen and paper-based exams to online and paperless mode.
* Candidates can appear for the exam using any desktop, laptop, or mobile device with a browser. Exam results can be generated instantly for the objective type of questions.
* It will provide details regarding student performance in previous papers on finger tips.

1.4.2 Features of proposed system

* It is time efficient.
* No extra human effort for attendance of students.
* Space compatibility.
* It saves paper and money.
* Almost null chances of misplacement of response sheets.
* The results will be auto-generated sharply after culmination of exams.
* Relieves the burden of manually checking the answer sheets.

**1.5 SCOPE OF PROJECT:**

The project covers a wide scope. The information of all the students can be stored. The main objectives of the project are as follows:

* To achieve efficiency in online examination.
* To save time of examiner and the examinee.
* To provide better security and transparency in the examination.
* To provide complete online web based solution, including student registration, giving tests and storing of results.
* To provide the results instantly.
* To reduce the administrator’s burden.
* To make evaluation massive but simple and cost-effective.
* To provide comfortable exam environment.
* To provide an easy to use web application.

Chapter 2

**AGILE APPROACH**

**2.1 Methodology**

Software engineering is a process of discovery, refinement, modeling and specification (a set of activities that is often referred to as analysis model of required data, information and control flow) for the creation of operational behavior. Alternative solutions are analyzed and a complete analysis model is created. Software requirements engineering process is described in the following way:

"Requirements engineering is the systematic use of proven principles languages and tools for the cost-effective analysis, documentation and ongoing evolution of user needs and the specification of external behavior of a system to satisfy those needs."

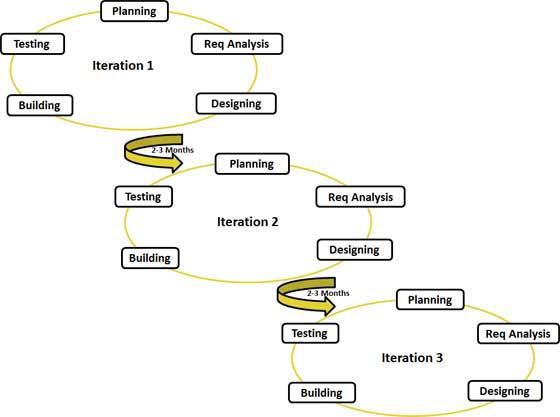
Both the software engineers and customers take an active role in the software requirement analysis. The customer attempts to reformulate sometimes-nebulous system level description of data, function and behavior into concrete detail. The developer acts as interrogator, consultant, or Requirement analysis are software engineering task that bridges the gap between the system level requirements engineering and software design. Requirements Engineering activities result in the specification of the software’s operational characteristics (data, function and behavior) indicates software interface with other system elements and establishes constraints that software must need. Requirement analysis allows the analyst to refine the software allocation and build models of data, functional and behavioral domains that will be treated by software. Requirement analysis provides the software designer with a representation of information, Function and behavior that can be translated to data, architectural, interface, and component level.

**2.2 Agile Approach:**

Agile development methodology provides opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular cadences of work, known as sprints or iterations, at the end of which teams must present a potentially shippable product increment. By focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology is described as “iterative” and “incremental.” In waterfall, development teams only have one chance to get each aspect of a project right.

In an agile paradigm, every aspect of development requirement, design, etc. is continually revisited throughout the lifecycle. When a team stops and re-evaluates the direction of a project every two weeks, there's always time to steer it in another direction. The results of this "inspect-and-adapt" approach to development greatly reduce both development costs and time to market.

Because teams can develop software at the same time, they're gathering requirements, the phenomenon known as "analysis paralysis" is less likely to impede a team from making progress. Agile development methodology helps companies build the right product. Instead of committing to market a piece software that hasn't even been written yet, agile empowers teams to continuously re-plan their release to optimize its value throughout development, allowing them to be as competitive as possible in the marketplace. Development using an agile methodology preserves a product's critical market relevance and ensures a team's work doesn't wind up on a shelf, never released.



**ITERATIONS IN AGILE MODEL:**

**2.3 Analysis and Design**

**2.3.1 Analysis Phase**

At this stage information about different advertising methodologies used by the employers have been obtained along with what means of information users find most feasible by interviewing. Project planning is done to ensure that the project should complete in budget and scheduled time.

This interviewing session was to get an insight into the problem domain, what solution people expect from our system, how we can make the information flow time and cost efficient from employer and job seeker. The insight gathered in the Requirements phase is elaborated to establish the baseline of the architecture of the system and provide a stable basis for the bulk of the development effort in the next phase.

A written document is produced to elaborate on requirement specifications in a more understandable form. Main focus is on "What" is to be developed rather "how" it should be developed. To be able to effectively and successfully capture and maintain requirements, ascertain understanding of the context of the system to be developed was gained. One approach to investigate the context of the system was done using Domain Modeling. This technique focused on things that existed or played a certain role in the context of the system.

The domain entities are then incorporated and structured in a UML class diagram, showing the relationships among the individual elements. Although, the analysis phase greatly resembles the Preliminary Investigation phase, any activities previously executed in the preliminary Investigation phase will now be executed in more depth. The completion of the Preliminary Investigation phase narrows the scope of the activities in the Analysis phase so that efforts will focus on the chosen solution. Now the models for the current system and proposed system can be fleshed out with more detailed specifications. Because the steering committee has approved the new system, work can begin in earnest.

**The Analysis phase has six basic activities:**

* Study the existing system.
* Review the conclusions obtained by the preliminary Investigation phase recommended solution, feasibility issues, and rejection of alternative solutions.
* Prepare the model of the new system.
* Revise the preliminary design.
* Devise the detailed schedule for project implementation.
* Prepare the report on the Analysis phase for review by management.
  + 1. **Preliminary investigation:**

This phase commences with discussion on the request made by the user. The request can be for a new system or for modifying the existing one. An estimate is made of whether the identified users’ needs can be satisfied or not. Preliminary investigation verifies the problem and understands the need for the required system. It considers whether the proposed system will be cost effective from the business point of view and it can be developed within existing budgetary constraints. In addition, the time factor, which determines the duration of the project is also considered. Preliminary investigation should be quick and cost effective. The output of preliminary investigation decides whether the new system should be developed or not.

* 1. **Feasibility Study**

A feasibility study looks at the viability of an idea with an emphasis on identifying potential problems and attempts to answer one main question: Will the idea work and should you proceed with it?

Feasibility studies address things like where and how the system will operate. They provide in-depth details about the system to determine if and how it can succeed, and serve as a valuable tool for developing a winning system plan. A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the product.

**2.4.1**  **Technical Feasibility:**

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may vary considerably but might include:

* The facility to produce outputs in a given time
* Response time under certain conditions.
* Ability to process a certain volume of transaction at a particular speed.
* Facility to communicate data to distant locations.

**2.4.2 Economic Feasibility:**

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as Cost/ Benefit analysis, the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs.

* + 1. **Operational Feasibility**:

Operational feasibility study tests the operational scope of the software to be developed. The points to be considered are:

* What changes will be brought with the system?
* What organizational structure are disturbed?
* What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?

**2.5 Design Phase:**

A software design is a meaningful engineering representation of some software product that is to be built. A design can be traced to the customer's requirements and can be assessed for quality against predefined criteria. In the software engineering context, design focuses on four major areas of concern: data, architecture, interfaces and components. The emphasis in design is on quality; this phase provides us with representation of software that can be assessed for quality. Furthermore, this is the only phase in which the customer's requirements can be accurately translated into a finished software product or system. As such software design serves as the foundation for all software engineering steps that follow regardless of which process model is being employed. Without a proper design we risk building an unstable system - one that will fail when small changes are made, one that may be difficult to test, one whose quality cannot be assessed until late in the software process, perhaps when critical deadlines are approaching and much capital has already been invested into the product. During the design process the software specification are transformed into design models that describe the details of the data structures, system architecture, interface and components. Each design product is reviewed for quality before moving to the next phase development. At the end of the design process a design specification document is produced. This document is composed of the design models that describe the data, architecture, interfaces and components.

Chapter 3

**ER DIAGRAM**

1. **ER Diagram**

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system’s entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity Core infrastructure.

**3.1. The elements of an ERD are:**

**1. Entities:** Entity is a real-world thing. It can be a person, place or even a concept. E.g., Department, Employee, Student, Courses, etc.

**2. Relationships:** An entity contains a real-world property called attribute. This is the characteristic of the attribute. E.g., the entity Employee has the property like Dept-Code, Emp-Id, Emp-Name, etc.

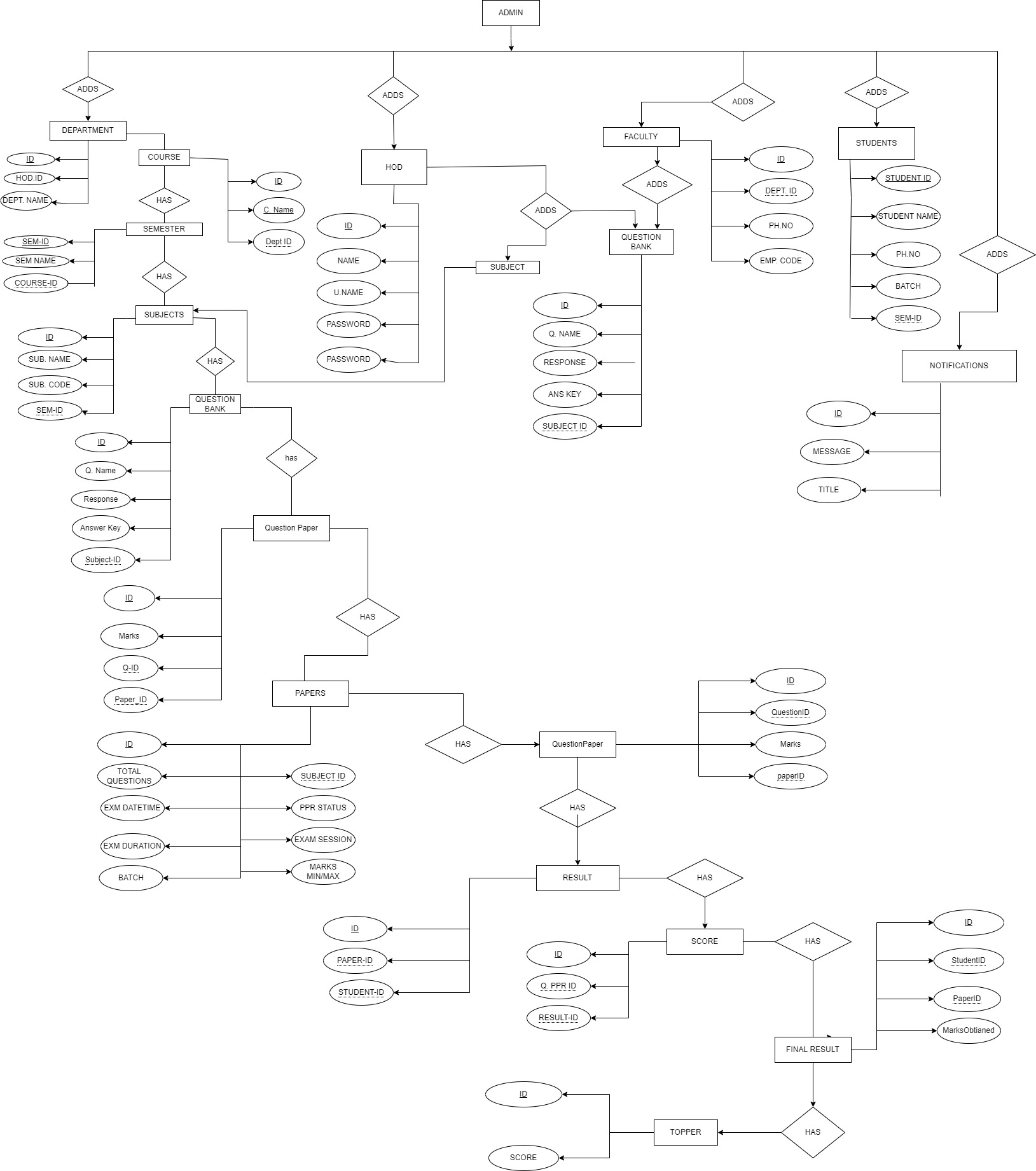
**3. Attributes:** relationship tells how two attributes are related. E.g., Employee works for Department.

Steps involved in creating an ERD include:

* Identifying and defining the entities
* Determining all interactions between the entities
* Analyzing the nature of interactions/determining the cardinality of the relationships
* Creating the ERD.

**3.2 E-R Diagram Components:**

* Rectangles represent entity sets.
* Ellipses represent attributes.
* Diamonds represent relationship sets.
* Lines link attributes to entity sets and entity sets to relationship sets.
* Double ellipses represent multivalued attributes.
* Dashed ellipses denote derived attributes.
* Primary key attributes are underlined



**Fig: 3.3 ER DIAGRAM**

Chapter 4

**DATA FLOW DIAGRAMS**

1. **Data Flow Diagrams**

A Data Flow Diagram (DFD) is a graphical tool used to describe and analyze the movement of data through a system – manual or automated including the processes, stores of data and delays in the system. They are central tools and the basis from which other components are developed. It depicts the transformation of data from input to output through processes and the interaction between processes. Transformation of data from input to output through processes logically and independent of physical components is called the DFD. The physical DFD shows the actual implementation and movement of data between people, departments and workstation.

DFD’s are an excellent mechanism of communicating with the customers during requirement analysis and are widely used for representing external and top-level internal design specification. In the Later situations, DFD’s are quite valuable for establishing naming conventions and names of system components such as subsystems, files and data links.

**In a DFD there are four components:**

* 1. Sources or Destinations of data such as human, entities that interact with system, outside the system boundary, who form the source and the recipient of information are depicted in the form of a closed rectangle.
  2. Data flow is a packet of data. It identifies data flow. It is a pipeline through which information flows. It is depicted in DFD as an arrow with the pointer pointing in the direction of flow. This connecting symbol connects an entity, process and data stores. This arrow mark also specifies the sender and the receiver.
  3. Process depicts procedure, function or module that transforms input data into output data. It is represented as a circle or a bubble with the procedure name and a unique number inside the circle.
  4. Data stores are the physical areas in the computer’s hard disk where a group of related data is stored in the form of files. They are depicted as an open-ended rectangle. The Data store is used either for storing data into the files or for reference purpose.

Student

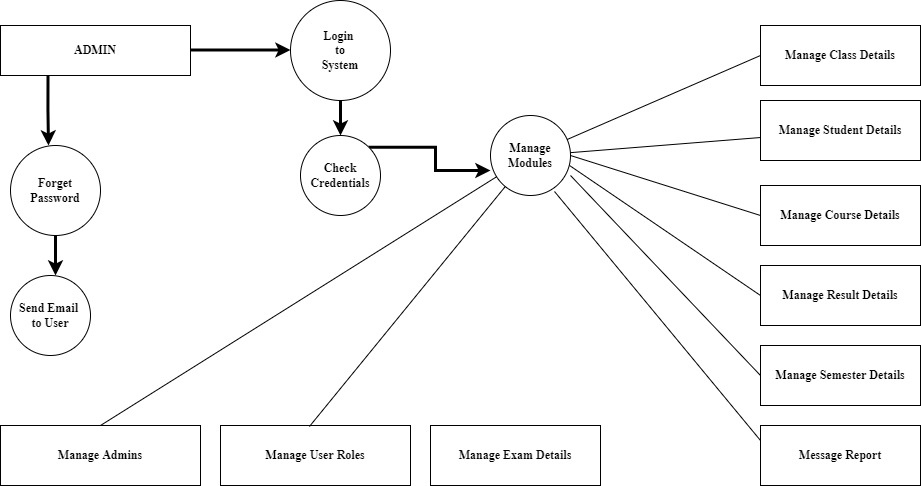
Faculty

HOD

Admin

**Database**

**Fig:** 4.1  **0-Level DFD**



**Fig:** 4.2  **1-Level DFD**

Chapter 5

**USE CASE DIAGRAMS**

**5. Use Case Models**

This model mainly depicts the user’s work in the system. These use cases provide comprehensive documentation under study and the responsibility of the system to its users.

**FIGURE: 5.1 USE CASE DIAGRAM FOR ADMIN**

**FIGURE: 5.2 USE CASE DIAGRAM FOR STUDENT**

**FIGURE: 5.3 USE CASE DIAGRAM FOR HOD**

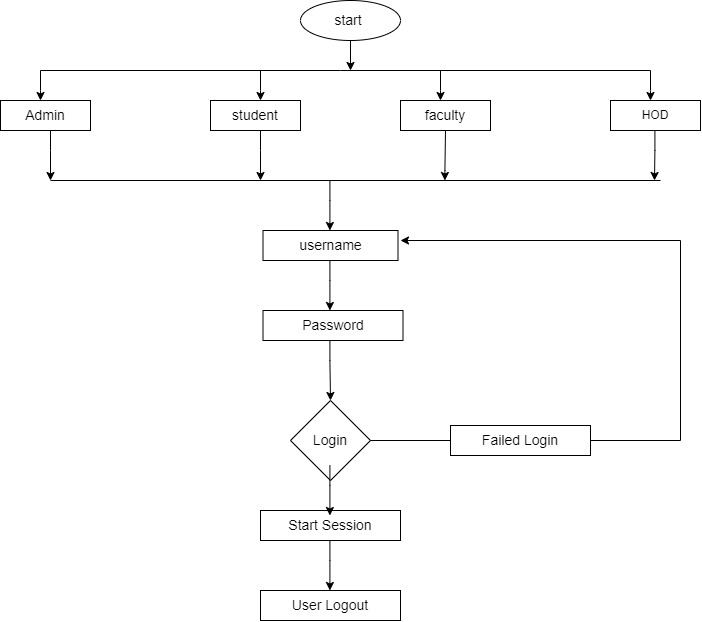
**FIGURE: 5.4 USE CASE DIAGRAM FOR FACULTY**

Chapter 6

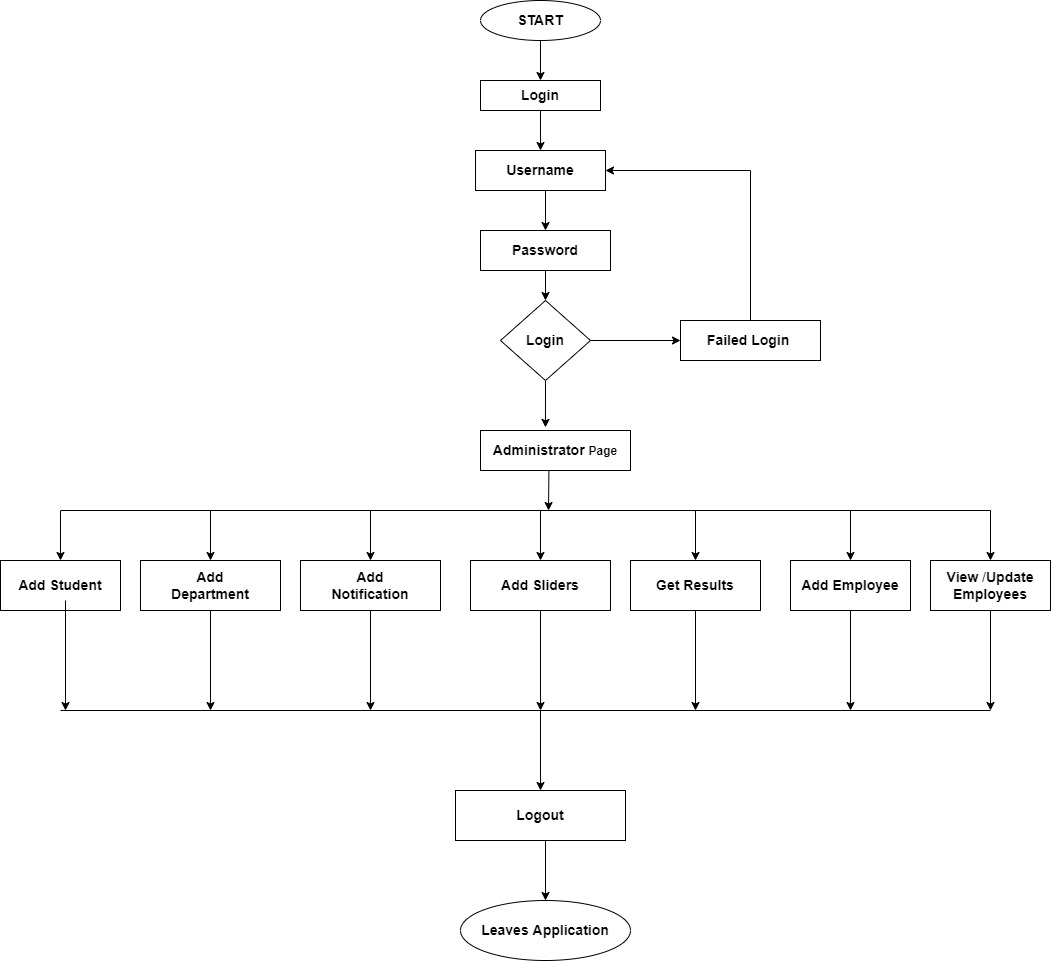
**ACTIVITY DIAGRAMS**

**6. Activity Diagrams**:

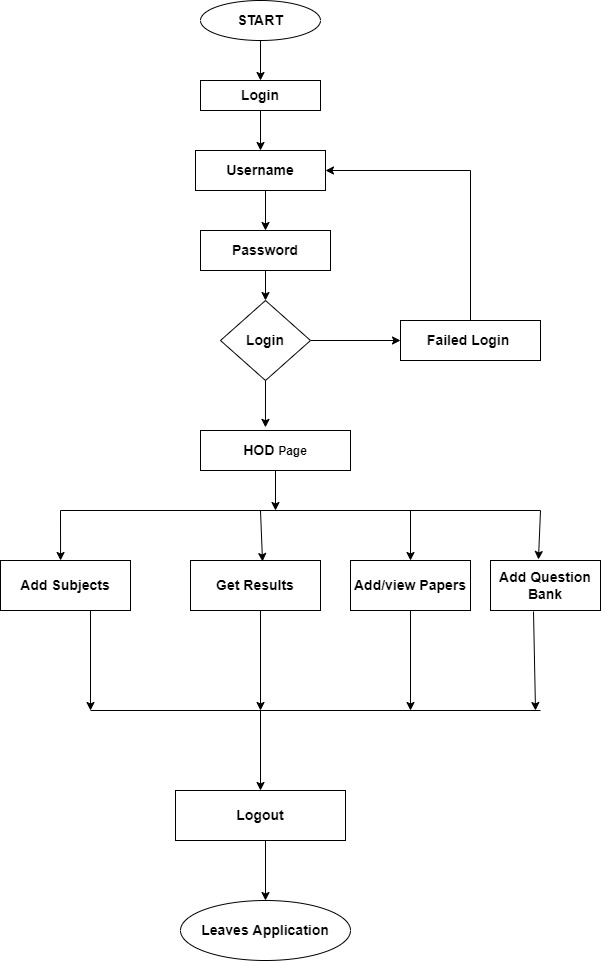
These Diagrams depict the activities that have been defined as part of the elicitation task. These function exist within a processing context. That is, the sequence of activities that describes processing within a limited context. Thus activity diagram describes the behavioral view of a system

****

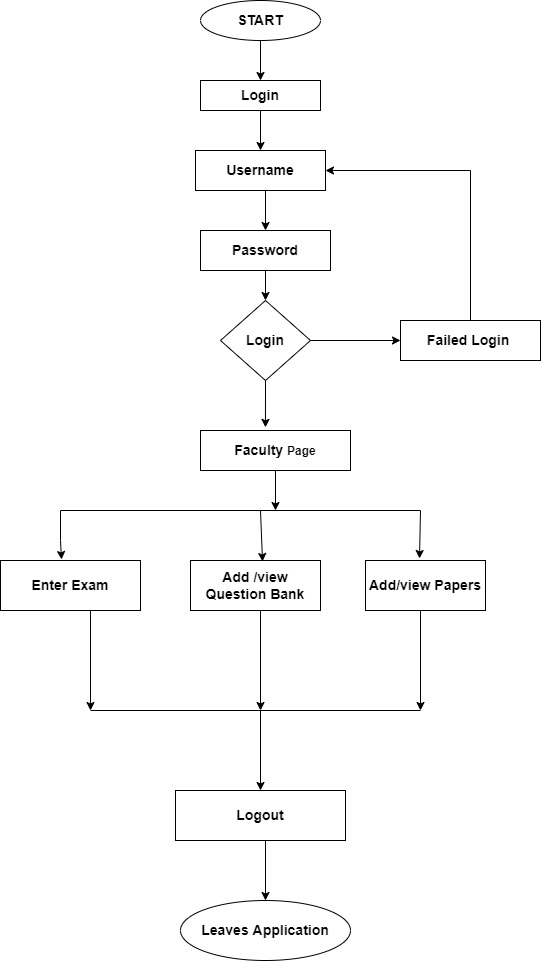
**Fig:** **6.1 Activity diagram for login**

****

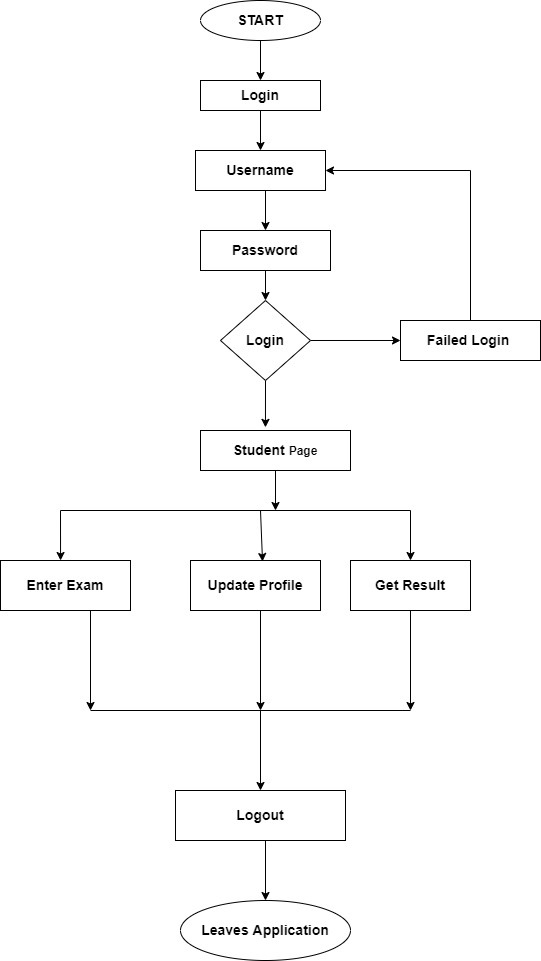
**Fig:** **6.2 Activity diagram for Admin**

****

**Fig:** **6.3 Activity diagram for HOD**



**Fig:** **6.4 Activity diagram for Faculty**

****

**Fig:** **6.5 Activity diagram for Student**

Chapter 7

**OVERVIEW OF**

**TECHNOLOGIES**

* 1. **overview of the technology:**

**7.1.1 Front end:**

* HTML:
* CSS:
* JavaScript /jQuery

HTML, is used to describe the structure of information on a web page. Together, HTML, CSS, and JavaScript make up the essential building blocks of websites, with CSS controlling a page’s appearance, and JavaScript programming its functionality.

* Bootstrap:

Bootstrap is a free and open-source collection of tools for Web Development it contains HTML, CSS, based design templates for typography, Forms, Buttons, navigation and other interface components, as well as optional JavaScript extensions. Bootstrap’s responsive CSS adjust to phones, tablets, and desktops. In Bootstrap3, mobile-first style is part of the core Core .it is compatible with all modern browsers (Chrome, Firefox, and Opera)

**7.1.2 Architecture:**

**Layered**

* Presentation Layer (WebApp MVC)
* Model Layer
* Business Layer
* Data Layer

**7.1.3 Back end:**

* ASP.NET MVC Core 3.1
* Sql server 2019
* C#

**7.1.4 Other Cores:**

* Entity Core 6.x,
* LINQ

**7.2 SYSTEM SPECIFICATION**

**7.2.1 HARDWARE REQUIREMENTS**

This section will describe the hardware requirement which is at least necessary to run this website.

**7.2.2** **PROCESSOR**

The minimum requirement of processor is 1.6GHz or faster processing speed. The speed of processor determines the time taken for the execution of the instructions. Higher processing speed leads to faster execution of the instruction.

**7.2.3 RAM**

RAM There should be minimum 1 GB of RAM available for the smooth functioning of the project. Higher memory leads in better results

**7.2.4 HARD DISK**

There should be about minimum 40GB of hard disk for smooth functioning of the project and so the recommended hard disk space is 40 GB.

**7.2.5**  **CACHE**

There must be about 512 KB of cache memory so that the accessing time can be better and compilation is easier.

The hardware used for the development of the project is:

PROCESSOR: Core I5 2.0 GHz

RAM: 8 GB

SSD: 256 GB

**7.2.6**  **SOFTWARE REQUIREMENTS**

The software used for the development of the project is:

OPERATING SYSTEM : Windows 10, 11, Mac OS, Linux

CODE EDITOR : Visual Studio

FRONT END : HTML 5, CSS3, Javascript

BACK END : ASP.NET MVC Core 3.1, Sql server 2019 ,C#

**7.2.7**  **MANPOWER REQUIREMENTS**

Project Duration: 6 Months

Team Size : 3

**7.3** **LANGUAGE SPECIFICATION:**

**7.3.1 C#:**

C# is an object-oriented, ***component-oriented*** programming language. C# provides language constructs to directly support these concepts, making C# a natural language in which to create and use software components. Since its origin, C# has added features to support new workloads and emerging software design practices. At its core, C# is an ***object-oriented*** language. You define types and their behavior. Several C# features help create robust and durable applications. Garbage collection automatically reclaims memory occupied by unreachable unused objects. Nullable types guard against variables that don't refer to allocated objects. Exception handling provides a structured and extensible approach to error detection and recovery. Lambda expressions support functional programming techniques. Language Integrated Query (LINQ) syntax creates a common pattern for working with data from any source. Language support for asynchronous operations provides syntax for building distributed systems. C# has a unified type system. C# emphasizes versioning to ensure programs and libraries can evolve over time in a compatible manner. Aspects of C#'s design that were directly influenced by versioning considerations include the separate virtual and override modifiers, the rules for method overload resolution, and support for explicit interface member declarations.

**7.3.2 CHARACTERISTICS OF C#:**

1. **Pure Object programming language (OOPS):**

C# and .Net are entirely based on the object-oriented programming system.

C# supports three object-oriented features.

* Encapsulation
* Inheritance
* Polymorphism

1. **Compatible with other Language:**

C# enforces .Net Core development and execution environment that allows different programming languages and library to work together to create window-based applications that are easier to build, manage, deploy and integrate with another new system

1. **Complete package:**

We can develop a different type of application using C#. C# helps to development Windows application, console application and web application with extra advanced Cores.

**4. Modern C# programming:**

C# based on current trends. It is very interactive and powerful in current time. It helps to develop a dynamic web page, XML web services, distributed application, a classical window desktop application, new smart client application, scalable, robust applications.

**5. Scalable & Updateable:**

In modern time, we update our applications. C# is a scalable and updateable programming language. To scale and update our application we delete the old files and update the them to new once. It's another importable and advance feature of C# programming**.**

Since the time of C# inception with .Net Core 6, the language is continuously being subjected to revisions in coordination with .NET updates to now become one of the most popular programming languages that creates Windows programs as well as web applications. Although, to an extent C# is a derivative of C and C++ and naturally has similar operators, object oriented case sensitivity, identical syntax etc., there are some major distinctions that sets this multi-paradigm language apart from all its applied languages. C# was designed for specific Microsoft's .Net Core the developers have access to not only feature of C# itself but to all the features of .Net application programming interface (API).

**7.4 .Net Core:**

.Net Core - pronounced and often written as dot net – the Microsoft developed software Core to principally run on Microsoft applications. The .Net is central component to MS overall development strategies as well as its Java auxiliary in MS development environment. Its huge standard class library called Core class library (FCL) enables wide ranging language interoperability among various programming languages where all the languages are compatible with codes written in other languages and yet be referred as managed code.

For the .Net Core performs in software exclusive environment like application virtual machine called common language runtime (CLR) – MS implementation of common language infrastructure (CLI) - that along with FCL sustain the whole platform. While CLR facilitates security, exception handling, thread execution, and memory management services, the FCL enables development of network communications, web application, and numeric algorithms by providing database connectivity, data accessibility, and user interface. The MS has also created an integrated development environment called Visual Studio for Windows applications that are developed by aligning the .Net Core supported code with various libraries.

## **7.4.1 Features of the common language runtime**

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application. The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally featuring rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers.

## **7.4.2 NET CORE CLASS LIBRARY**

The .NET Core class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Core types easy to use, but also reduces the time associated with learning new features of the .NET Core. In addition, third-party components can integrate seamlessly with classes in the .NET Core.

For example, the .NET Core collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Core.

As you would expect from an object-oriented class library, the .NET Core types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Core to develop the following types of applications and services:

* Console applications
* Scripted or hosted applications.
* Windows GUI applications (Windows Forms).
* ASP.NET applications.
* XML Web services.
* Windows services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes

**7.4.3 Principal Design Features of .Net Core:**

The primary objective of .Net Core design is to increase the productivity and minimize the margin of programming errors through modular approach integration into the software structure. **.**Net Cores has number of in-built tools that ensure easy program deployment - installing to execution - without disrupting the environment created by formerly installed applications. **.**Net security is known for its compelling solutions that are based on a transparent control security model. .Net developed software applications are put to through specific model for the code behavior monitoring in all circumstances for the purpose of risk mitigation in applications including client, web, and server-side.

**7.4.4 .NET Core working:**

The .Net Core is a concentrated developer platform which is comprised of several tools, libraries, programming languages, and features to develop widely varying applications. It works by compiling the source code of one language in an intermediate language (IL) that is usually put in disk storage in a file called assembly which can be executed Since .Net, MS developed programming infrastructure to develop, deploy, and run multiple services and applications e.g., desktop and web app that are based on .Net technologies, the assembly file contains directions of information for security and type of used code. After the assembly file is uploaded and validated by the CLR, the IL compiled code is interpreted to become instructions written in native machine code.

The .Net Core utilizes three principal process fragment that include:

* Common language runtime
* Core class library
* ASP.Net

.NET Core led to a grand family of multiple .NET platforms e.g., Net, .Net Core, and .Net core. The .Net Core is consisted of various implementations that allow codes developed in this MS environment to execute in number of different platforms such as, Linux, macOS, Windows, iOS, Android etc. Following are some of the most frequently used .Net implementations that include:

1. **.Net Framework**  is the primary implementation of whole .Net family which is used to support websites functionality, web & desktop app development, delivery of numerous IT services, and much more on Windows operating systems.
2. **.NET Core** is an open source cross-platform implementation that also supports websites functions, IT services along with the development of console apps on not only Windows, but also on Linux and macOS.
3. **.NET Standard** is a not an implementation itself but is more of an API  
   specification that lies across all .Net implementations for the purpose of  
   facilitating the usability of the same code and libraries to smoothly run on  
   number of different implementations.

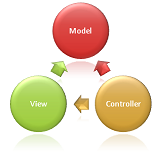
**7.4.5 ASP.NET Core MVC:**

ASP.NET Core MVC is a rich Core for building web apps and APIs using the Model-View-Controller design pattern.

**MVC pattern:**

The Model-View-Controller (MVC) architectural pattern separates an application into three main groups of components: Models, Views, and Controllers. This pattern helps to achieve separation of concerns. Using this pattern, user requests are routed to a Controller which is responsible for working with the Model to perform user actions and/or retrieve results of queries. The Controller chooses the View to display to the user, and provides it with any Model data it requires.

The following diagram shows the three main components and which ones reference the others:



This depiction of responsibilities helps you scale the application in terms of complexity because it's easier to code, debug, and test something (model, view, or controller) that has a single job. It's more difficult to update, test, and debug code that has dependencies spread across two or more of these three areas. For example, user interface logic tends to change more frequently than business logic. If presentation code and business logic are combined in a single object, an object containing business logic must be modified every time the user interface is changed. This often introduces errors and requires the retesting of business logic after every minimal user interface change.

**Model Responsibilities:**

The Model in an MVC application represents the state of the application and any business logic or operations that should be performed by it. Business logic should be encapsulated in the model, along with any implementation logic for persisting the state of the application. Strongly-typed views typically use ViewModel types designed to contain the data to display on that view. The controller creates and populates these ViewModel instances from the model.

**View Responsibilities:**

Views are responsible for presenting content through the user interface. They use the Razor view engine to embed .NET code in HTML markup. There should be minimal logic within views, and any logic in them should relate to presenting content. If you find the need to perform a great deal of logic in view files in order to display data from a complex model, consider using a View Component, ViewModel, or view template to simplify the view.

**Controller Responsibilities:**

Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render. In an MVC application, the view only displays information; the controller handles and responds to user input and interaction. In the MVC pattern, the controller is the initial entry point, and is responsible for selecting which model types to work with and which view to render (hence its name - it controls how the app responds to a given request).

**7.5 Advantages of C# & .Net programming**

* C# being object-oriented language allows modular application development and code reusability.
* The CLR integration makes for C# programs’ smooth assimilation with code component from other languages.
* The .Net Core support make C# programming interpretable among all .Net technologies.
* Due to the formal all-set method approach, the C# coding becomes significantly more intelligible and eliminates the need to be concerned about header files.
* The massive choice of Microsoft stack and rich library, the C# provides tool for just about every function and implementation which allow programmers to freely experiment with new compositions and code compilation.
* The .Net Core provides in-house MS support for C# programming which is quite unlike other languages where programmers are solely dependent on communities for support.
* Backward compatibility arises from Microsoft continuous discarding of old Window platform which requires .Net Core upgrade. The externally instilled need to incessantly improve the configuration serves to give user afresh motivation to work hard.

**7.6 SQL SERVER 2019**

Microsoft SQL Server (pronounced as “sequel server”) is one of the most popular databases used by many companies for decades. It also goes with other names like MSSQL, MS SQL Server, or simply, SQL Server. These databases are used to store relevant information to run a business. It is Microsoft’s flagship database. It runs on Windows, Linux, and Mac (through Docker). You can also use it in the cloud through Azure, AWS, and Google Cloud.

SQL Server 2019 (15.x) builds on previous releases to grow SQL Server as a platform that gives you choices of development languages, data types, on-premises or cloud environments, and operating systems. SQL Server 2019 (15.x) introduces Big Data Clusters for SQL Server. It also provides additional capability and improvements for the SQL Server database engine, SQL Server Analysis Services, SQL Server Machine Learning Services, SQL Server on Linux, and SQL Server Master Data Services. Microsoft SQL Server 2019 powers your organization by providing a data hub that you can use to access structured and unstructured data sources from across your entire data estate through a consistent interface. The relational database engine scales to petabytes of data, and enhancements to PolyBase allow you to process diverse big data and relational data sources using Transact-SQL from SQL Server.

**7.6.1 Features of Sql Server 2019**

## **Intelligent Query Processing:** [Intelligent Query Processing](https://geopits.com/sql-server/versions/2017/#rdbms-scalability-and-performance) in SQL server 2019 version improves the performance of existing workloads with minimal implementation effort. Built on intelligent query processing features of SQL 2017, the SQL 2019 version has its own set of capabilities and improvements that can meet all your data needs.

1. **Accelerated Database Recovery (ADR):**

SQL Server 2019 ADR feature helps improve database availability, especially for long-running data transactions. With accelerated database recovery (ADR):

* Long-running transactions can roll back instantly, irrespective of the time that the transaction has been active or the number of updates that it has performed. ADR uses Persisted Version Store for tracking changes.
* Time taken by long-running transactions does not affect the recovery time, enabling faster database recovery irrespective of the size of active transactions in the system.
* The transaction log is shortened even when long-running transactions are active, keeping it in control.

1. **Memory-Optimized Tempdb Metadata:**

Heavy workloads on Tempdb have always been a challenge. Microsoft has over the years made a lot of improvements to SQL Server to overcome this shortcoming. Tempdb system table metadatais one where too many sessions are trying to write to system tables.With SQL Server 2019, you can now move over ten of the most-used Tempdb’s system tables into memory-optimized tables. It helps with resolving most of the issues in the query, thereby making TempDB more scalable.

1. **Replication in Linux:**

SQL Server on Linux now supports replication, including Snapshot, Transactional, and even Merge. The instance can support any of the replication roles: Publisher, Distributor, Subscriber. You can even go wild and mix and match the operating systems involved! For example, the publisher and distributor can be on Windows or Linux, and the subscriptions can be on a mix of Windows and Linux instances as well.

**7.6.2 Advantages of Sql Server 2019**

### **Easy Installation**

All the Microsoft products are easy to install with the one-click installation procedure and readable GUI with lots of instructions for the layman. MS SQL Server contains all these characteristics and it was an extremely user-friendly installation interface, unlike other database servers that require extensive command-line configurations. Principally to download MS SQL Server, you require a net Core, a minimum of 1GB memory, and NTFS system.

1. **Improved Performance**

MS [SQL server contains](https://bytescout.com/blog/simple-sql-database.html?utm_referer=https%3A%2F%2Fwww.google.com%2F) excellent compression and encryption capabilities that result in improved data storage and retrieval functions.

1. **Security**

MS SQL server is considered one of the most secure database servers with complex encryption algorithms making it virtually impossible to crack the security layers enforced by the user. MS SQL server is not an open-source [database](https://bytescout.com/blog/2014/10/mysql-server-history-and-advantages.html?utm_referer=https%3A%2F%2Fwww.google.com%2F) server which reduces the risk of attacks on the database server.

1. **Excellent Data Restoration and Recovery Mechanism**

**MS SQL server is fully aware of the importance of your data. Hence MS SQL Server contains many sophisticated features that allow you to recover and restore the data which has been lost or damaged.** Although you cannot recover individual records but it is possible to restore a complete database using some advanced recovery tools contained in the MS SQL Server database.

**7.7 ENTITY CORE 6.x:**

Entity Core 6 (EF6) is a tried and tested object-relational mapper (O/RM) for .NET with many years of feature development and stabilization. As an O/RM, EC6 reduces the impedance mismatch between the relational and object-oriented worlds, enabling developers to write applications that interact with data stored in relational databases using strongly-typed .NET objects that represent the application's domain, and eliminating the need for a large portion of the data access "plumbing" code that they usually need to write.

**7.8 Language Integrated Query (LINQ) (C#):**

Language-Integrated Query (LINQ) is the name for a set of technologies based on the integration of query capabilities directly into the C# language. Traditionally, queries against data are expressed as simple strings without type checking at compile time or IntelliSense support. Furthermore, you have to learn a different query language for each type of data source: SQL databases, XML documents, various Web services, and so on. With LINQ, a query is a first-class language construct, just like classes, methods, events. You write queries against strongly typed collections of objects by using language keywords and familiar operators. The LINQ family of technologies provides a consistent query experience for objects (LINQ to Objects), relational databases (LINQ to SQL), and XML (LINQ to XML).

For a developer who writes queries, the most visible "language-integrated" part of LINQ is the query expression. Query expressions are written in a declarative query syntax. By using query syntax, you can perform filtering, ordering, and grouping operations on data sources with a minimum of code. You use the same basic query expression patterns to query and transform data in SQL databases, ADO.NET Datasets, XML documents and streams, and .NET collections.

***7.9 HTML:***

To understand "HTML" from front to back, let's look at each word that makes up the abbreviation:

**Hypertext**: text (often with embeds such as images, too) that is organized in order to connect related items

**Markup**: a style guide for typesetting anything to be printed in hardcopy or soft copy format

**Language**: a language that a computer system understands and uses to interpret commands.

HTML determines the structure of web pages. This structure alone is not enough to make a web page look good and interactive. So you'll use assisted technologies such as CSS and JavaScript to make your HTML beautiful and add interactivity, respectively.

In this case, I like to break down the three technologies – HTML, CSS, and JavaScript – this way: they are like a human body.

* HTML is the skeleton,
* CSS is the skin,
* and JavaScript is the circulatory, digestive, and respiratory systems that brings the structure and the skin to life.

*.*

***7*.9.1Applications of HTML**

* ***Web page development****-* HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.
* **Internet Navigation** - HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
* **Responsive UI** - HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive designstrategy.
* **Offline support** HTML pages once loaded can be made available offline on the machine without any need of internet.
* **Game development**- HTML5 has native support for rich experience and is now useful in gaming development arena as well.

**7.10 CSS:**

CSS stands for **Cascading Style Sheet.** It gives an additional style to the HTML document. A cascading style sheet is a language that is designed to define the document formatting and look written in a markup language. Generally, CSS is applied with HTML documents to change various styles of user interfaces and web pages.

* CSS was introduced on **10th October 1994** by **Hakon Wium Lie***.*
* CSS is an easy language to understand and learn, but it gives robust control on the presentation of HTML documents
* It specifies how the various HTML elements or tags to be presented on the screen.
* It has various advantages, such as save time, offline browsing, fast page load, and many others.
* CSS3 is the current version of the cascading style sheet.

It can be also applied with any document of XML, including XUL, SVG, and plain XML.

### Uses of CSS

* We can add unique styles to our old documents of HTML.
* We can change the overall look and feel of our website by following some changes in the CSS code.
* A cascading style sheet can be used with JavaScript and HTML in most of the websites to develop user interfaces for a lot of mobile applications and user interfaces for various web applications.

### **7.10.1 Features of CSS:**

* **Opportunity in Web designing:**If anyone wants to begin a career in web designing professionally, it is essential to have knowledge of CSS and HTML.
* **Website Design:**With the use of CSS, we can control various styles, such as the text color, the font style, the spacing among paragraphs, column size and layout, background color and images, design of the layout, display variations for distinct screens and device sizes, and many other effects as well.
* **Web Control:**CSS has controlling power on the documents of HTML, so it is easy to learn. It is integrated with the HTML and the XHTML markup languages.
* **Other Languages:**Once we have knowledge of some basics of CSS and HTML, other associated technologies like Angular, PHP, and JavaScript are become clearer to understand.

### **7.10.2 CSS Applications:**

Some critical applications of CSS are discussed and listed below:

* **Fast Page Loading:**We don’t need to mention the attributes of the HTML element every time if we use CSS. We need to specify one rule of CSS for an element and use it for every occurrence of that element.  So, short code means high-speed download times.
* **Easy Maintenance:**To create a global alteration we need to alter the style. Every element will get automatically updated within all web pages.
* **Superior HTML Styles:**HTML contains few extended attribute’s array than CSS, thus we can provide a much better view to our HTML page as compared to HTML attributes.
* **Save Time:**We can specify CSS once and reuse the same sheet within various HTML pages. We can describe a style for every HTML tag and apply this style to as a lot of web pages as we want.
* **Compatibility:**The cascading style sheet permits content to be upgraded for one or more device types. Distinct versions of the website could be granted for various handheld devices like cell phones and PDAs, and for printing.
* **Global Standards:**Now, the attributes of HTML are being recommended to apply CSS, and they are being deprecated. Thus it is better to begin the use of CSS in every HTML page for making them compatible for future browsers*.*

**7.11 JAVASCRIPT:**

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities. JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm

**7.12 BOOTSTRAP:**

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript Core for developing responsive, mobile-first websites. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). All thanks to Bootstrap developers – Mark Otto and Jacob Thornton of Twitter, though it was later declared to be an open-source project.

**Why we use Bootstrap?**

* It is Faster and Easier way for Web-Development.
* It creates Platform-independent web-pages.
* It creates Responsive Web-pages.
* It designs the responsive web pages for mobile devices too.

Chapter 8

**DATABASE DESIGN**

**8. Database Design:**

The Project has a MS SQL Server 2019 as its backend for database connectivity. The database stores the various information regarding the project like, user registration Blog posts, Articles and offline messages.

**MS SQL Server 2019:**

MS SQL Server 2019 is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those running on the same or another computer across a network. Its primary query language is T-SOL & ANSI SQL.

SQL server 2019 aims to make data management self-tuning, self- organizing and self- maintaining. The full text search functionality has been integrated with the database engine. This simplifies management& improves performance.

MS SQL server 2019 runs on windows 8/10/11. It is fully multi- threaded using kernel threads, and provides application program interfaces (APIs) for many Programming languages. MYSQL is used in a wide range of applications, including e- commerce, Web databases, logging applications and distributed applications.

**8.1. User Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | Unique identifier | Primary key | Id of the User |
| Name | nvarchar(max) | - | Name of the user |
| UserName | nvarchar(max) | - | Username by which  User can be differentiated |
| Password | nvarchar(max) | - | Encrypted Password |
| Email | nvarchar(max) | - | Contacting Email |
| Reset code |  | - | Code used to reset  password |
| Salt | nvarchar(max) | - | Used to Encrypt  Password |
| Image Path | nvarchar(max) | - | Physical Path of the  user’s display image |
| UserStatus | Int | - | Determine whether  user is active or not |
| UserRole | Int | - | Role Of a User |

**8.2 Department Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | Uniqueidentifier | Primary key | Id of the  Department |
| DepartmentName | Nvarchar(max) | - | Name of the  Department |
| HodId | Uniqueidentifier | Foreign key | Id of the HOD |

**8.3 Course Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | uniqueidentifier | Primary key | Id of the Course |
| CourseName | nvarchar(max) | - | Name of the Course |
| DepartmentID | uniqueidentifier | Foreign Key | Id of the  Department |

**8.4 Semester Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | uniqueidentifier | Primary key | Id of the Semester |
| Sem | nvarchar(max) | - | Name of the Semester |
| CourseID | uniqueIdentifier | Foreign key | Id of the Course |

**8.5 Student Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | Uniqueidentifier | Primary key,  Foreign key | Id of the Student |
| PhoneNo | nvarchar(max) | - | Phone number of the Student |
| CourseID | uniqueIdentifier | Foreign key | Id of the Course |
| RegistrationNo | nvarchar(max) | - | Registration Number of the Student |
| Batch | Int | - | Batch of the Student |

**8.6 Subject Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | Uniqueidentifier | Primary key, Foreign key | Id of the subject |
| PhoneNo | nvarchar(max) | - | Registration Number of the Student |
| RegistrationNo | nvarchar(max) | - | Registration Number of the Student |
| SemID | Uniqueidentifier | Foreign key | Id of the semester |
| Batch | Int | - | Batch of the Student |

**8.7 Question Bank Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | uniqueidentifier | Primary key | Id of the Question Bank |
| QuestionName | nvarchar(max) | - | Name Of the Question |
| Option1 | nvarchar(max) | - | Choice of the question |
| Option2 | nvarchar(max) | - | Choice of the question |
| Option3 | nvarchar(max) | - | Choice of the question |
| Option4 | nvarchar(max) | - | Choice of the question |
| Answerkey | nvarchar(max) | - | Correct choice of the Question |
| SubjectId | uniqueidentifier | Foreign key | Id of the Subject |

**8.8 Question Papers Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | Uniqueidentifier | Primary key | Id of the question papers |
| QuestionId | Uniqueidentifier | Foreign key | Id of the Question Bank |
| Marks | Int | - | Marks of the Questions |
| PaperId | Uniqueidentifier | Foreign key | Id of the Paper |

**8.9 Paper Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | uniqueidentifier | Primary key | Id of the question paper |
| TotalQuestions | Int | - | Total number of questions in paper. |
| ExamDateTime | datetime | - | Date and Time of the Examination |
| Examduration | Int | - | Total Duration of the Examination |
| MinimumMarks | Int | - | Minimum pass marks of the Examination |
| MaximumMarks | Int | - | Maximum marks of the Examination |
| PaperStatus | smallint | - | Active or Not |
| Batch | Int | - | Batch of the Student |
| ExamSession | Smallint | - | Session of the Examination |
| SubjectId | uniqueidentifier | Foreign key | Id of the Subject |

**8.10 Result Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | uniqueidentifier | Primary key | Id of the Result |
| PaperId | uniqueidentifier | Foreign key | Id of the Paper |
| StudenttId | uniqueidentifier | Foreign key | Id of the Student |

**8.11 Score Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | uniqueidentifier | Primary key | Id of the Score |
| ResultId | uniqueidentifier | Foreign key | Id of the Result |
| PaperQuestionId | uniqueidentifier | Foreign key | Id of the Question Paper |
| Answer | nvarchar(max) | - | Correct Answer of the Question |

**8.12 Final Result Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | uniqueidentifier | Primary key | Id of the Final Result |
| StudentId | uniqueidentifier | Foreign key | Id of the Student |
| PaperId | uniqueidentifier | Foreign key | Id of the Question Paper |
| MarksObtianed | Float | - | Total marks obtained in the Question paper |

**8.13 Topper Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Constraint | Description |
| ID | uniqueidentifier | Primary key | Id of the Topper |
| Score | Nvarchar(max) | - | Total Marks obtained by the Student |

Chapter 9

**TESTING**

**9. Testing Techniques**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing can be stated as the process of validating and verifying that a software program/application/product:

Meets the requirements that guided its design and development works as expected, and can be implemented with the same characteristics.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted.

Different software development models will focus the test effort at different points in the development process. Newer development models, such as Agile, often employ test driven development and place an increased portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been co Testing can never completely identify all the defects within software.

Instead, it furnishes a criticism or comparison that compares the state and behavior of the product against oracles—principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts, comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria.

Every software product has a target audience. For example, the audience for video game software is completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it can assess whether the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment.

**9.1 Design of Test cases**

Design of Test case is an important & challenging as software designing and implementation. The approaches followed here for the Designing of Test cases are Black-Box Testing and White-Box Testing. We had done both testing techniques (White Box Testing and Black Box Testing) to our application.

* + 1. **Black-box testing**

This testing looks at the available inputs for an application and the expected outputs that should result from each input. It does not have any relation with the inner workings of the application, the process undertaken or any other internal aspect of the application. Search engine is a very good example of a black box system. We enter the text that we want to search, by pressing “search” we get the results. Here we are not aware of the actual process that has been implemented to get the results. We simply provide the input and get the results.

* **Advantages of Black-box testing**

Since tester does not have to focus on the inner working of an application, creating test cases is easier.

Test case development is faster as tester need not to spend time on identifying the inner processes; his only focus is on the various paths that a user may take through GUI.

It is simple to use as it focuses only on valid and invalid inputs and ensures that correct outputs are obtained.

* + 1. **White-box testing:**

This testing looks into the complex inner working of the application; it tests the processes undertaken and other internal aspects of the application. While black box testing is mainly concerned with the inputs and outputs of the application, white box testing helps us to see beyond i.e., inside the application. White-box testing requires a degree of sophistication which is not the case with the black-box testing, as the tester is required to interact with the objects that are used to develop an application rather than having easy access to the user interface. In-circuit testing is a good example of a white-box system testing where the tester is looking at the interconnections between different components of the application and verifying the proper functioning of each internal connection. We can also consider the example of an auto-mechanic who takes care of the inner workings of a vehicle to ensure that all the components are working correctly to ensure the proper functioning of the vehicle.

* **Advantages of White-box testing**

Since the focus is on the inner working the tester can identify objects programmatically. This can be useful

when the GUI is frequently changing.

It can improve stability and re usability of test cases provided the object of an application remains the same.

By testing each path completely, it is possible for a tester to achieve thoroughness.

* 1. **Unit Testing**

Unit testing is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming

unit may be an individual function or procedure. In object-oriented programming a unit is usually an interface, such as a class.

Unit tests are created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others: substitutes like method stubs, mock objects, fakes and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits. Unit tests find problems early in the development cycle.

* 1. **Integration Testing:**

Integration testing (sometimes called Integration and Testing, abbreviated "I&T") is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using Black box testing, success and error cases being simulated via appropriate parameter and data inputs.

Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing.

The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

* 1. **System testing**

System testing tests the integration of each module in the system. It also tests to find discrepancies between the system and it’s original objective, current specification and system documentation. The primary concern is the compatibility of individual modules. Entire system is working properly or not will be tested here, and specified path ODBC connection will correct or not, and giving output or not are tested here these verifications and validations are done by giving input values to the system and by comparing with expected output. Top-down testing implementing here.

## **Acceptance Testing**

This testing is done to verify the readiness of the system for the implementation. Acceptance testing begins when the system is complete. Its purpose is to provide the end user with the confidence that the system is ready for use. It involves planning and execution of functional tests, performance tests and stress tests in order to demonstrate that the implemented system satisfies its requirements.

Tools to special importance during acceptance testing include:

Test coverage Analyzer – records the control paths followed for each test case.

Timing Analyzer – also called a profiler, reports the time spent in various regions of the code are areas to concentrate on to improve system performance.

Coding standards – static analyzers and standard checkers are used to inspect code for deviations from standards and guidelines.

Unit testing

Module testing

Sub-system testing

System testing

Acceptance testing

**Figure 9.**6 Testing Process

**9.7 Test Cases:**

Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.

Using White-Box testing methods, the software engineer can drive test cases that

* Guarantee that logical decisions on their true and false sides.
* Exercise all logical decisions on their true and false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structure to assure their validity.

The test case specification for system testing has to be submitted for review before system testing commences.

**9.7.1 Test Cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO. | Action | Input | Output | Actual Result |
| **LOGIN FORM** | | | | |
| 1 | If wrong user name or password is entered, message should be displayed | Any invalid username /password | An error message “invalid username or password “is displayed | An error message “invalid username or password “is displayed |
| 2 | If a valid username or password is entered, it should display the main menu of the user as specified in role | Username  Admin  Password  Admin | Welcome page and the main menu of the applicant | Welcome page and the main menu of the applicant |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO. | Action | Input | Output | Actual Result |
| **Adding new User** | | | | |
| 1 | If any username/password/role is left blank, an error message should be displayed | Try to leave any field blank. | An error message “Please enter specified field” | An error message “Please enter specified field” |
| 2 | If any username is entered that already exists, then error message should be displayed. | Any username which has been already registered is entered | An error message “Username already exists” should be displayed. | An error message “Username already exists” should be displayed. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO. | Action | Input | Output | Actual Result |
| **Changing Password** | | | | |
| 1 | If any username/password/role is left blank, an error message should be displayed | Try to leave any field blank. | An error message “Please enter specified field” | An error message “Please enter specified field” |
| 2 | If wrong user name or password is entered, message should be displayed | Invalid username or password | An error message “Invalid username/ password” should be displayed | An error message “Invalid username/ password” should be displayed |
| 3 | If new password and confirm password does not match. | Different new password and confirm password | Error message “Password does not match” should be displayed” | Error message “Password does not match” should be displayed” |

Chapter 10

**IMPLEMENTATION**

**10.1 Domain Models**

**10.1.1 Base Model**

public class BaseModel

{

public DateTime CreatedOn { get; set; }=DateTime.Now;

public Guid CreatedBy { get; set; }

public DateTime UpdatedOn { get; set; }

public Guid UpdatedBy { get; set; }

}

**10.1.2 User Model**

public class User : BaseModel

{

public Guid Id { get; set; }

public string? Name { get; set; }

public string? UserName { get; set; }

public string? Password { get; set; }

public string? Email { get; set; }

public string? ResetCode { get; set; }

public string? Salt { get; set; }

public string? ImagePath { get; set; }

public Status UserStatus { get; set; }

public UserRole UserRole { get; set; }

public Employee? Employee { get; set; }

public Student Student { get; set; }

}

**10.1.3 Course Model**

public class Course : BaseModel

{

public Guid? Id { get; set; }

public string? CourseName { get; set; }

public Guid? DepartmentId { get; set; }

[ForeignKey(nameof(DepartmentId))]

public Department Department { get; set; }

public ICollection<Semester> Semesters { get; set; }

}

**10.1.4 Department Model**

public class Department : BaseModel

{

public Guid Id { get; set; }

public string? DeptName { get; set; }

public Guid? HODId { get; set; }

public ICollection<Course> Courses { get; set; }

public ICollection<Employee> Employees { get; set; }

}

**10.1.5 Employee Model**

public class Employee : BaseModel

{

public Guid Id { get; set; }

[ForeignKey(nameof(Id))]

public User User { get; set; }

public string? PhoneNo { get; set; }

public string EmpCode { get; set; }

public Guid DepartmentId { get; set; }

[ForeignKey(nameof(DepartmentId))]

public Department Course { get; set; }

}

**10.1.6 Paper Model**

public class Paper : BaseModel , IRootModel

{

public Guid Id { get; set; }

public int TotalQuestion { get; set; }

public DateTime ExamDateTime { get; set; }

public int ExamDuration { get; set; }

public int MinimumMarks { get; set; }

public int MaximumMarks { get; set; }

public string RejectedReason { get; set; }

public PaperStatus PaperStatus { get; set; }

public int Batch { get; set; }

public Session ExamSession { get; set; }

public Guid SubjectId { get; set; }

[ForeignKey(nameof(SubjectId))]

public Subject subject { get; set; }

public ICollection<PaperQuestion> PaperQuestions { get; set; }

public ICollection<Result> Results { get; set; }

public ICollection<FinalResult> FinalResults { get; set; }

}

* 1. **Repository Pattern**

**10.2.1 IRepository**

public interface IRepository

{

IQueryable<T> GetAll<T>() where T : class;

T GetById<T>(Guid id) where T : class;

void DeleteRange<T>(List<Guid> ids) where T : class, IRootModel, new();

IQueryable<T> FindBy<T>(System.Linq.Expressions.Expression<Func<T, bool>> predicate) where T : class;

void AddList<T>(IEnumerable<T> obj) where T : class;

bool IsExist<T>(System.Linq.Expressions.Expression<Func<T, bool>> predicate) where T : class;

int Save();

int AddAndSave<T>(T model) where T : class;

int UpdateAndSave<T>(T model) where T : class;

int DeleteAndSave<T>(Guid id) where T : class, IRootModel, new();

IDbContextTransaction BeginTransaction();

IQueryable<T> FromQuery<T>(string sql, params object[] parameters) where T : class;

T GetObject<T>(string sql, params object[] parameters) where T : class;

int ExecuteQuery(string query, params object[] parameters);

}

**10.2.2 Repository**

public class Repository : IRepository

{

readonly AppDbContext dbContext;

public Repository(AppDbContext dbContext)

{

this.dbContext = dbContext;

}

public IQueryable<T> GetAll<T>() where T : class

{

return dbContext.Set<T>();

}

public T GetById<T>(Guid id) where T : class

{

return dbContext.Set<T>().Find(id);

}

public void DeleteRange<T>(List<Guid> ids) where T : class, IRootModel, new()

{

var list = new List<T>();

ids.ForEach(x => list.Add(new T { Id = x }));

dbContext.Set<T>().RemoveRange(list);

}

public IQueryable<T> FindBy<T>(System.Linq.Expressions.Expression<Func<T, bool>> predicate) where T : class

{

return dbContext.Set<T>().Where(predicate);

}

public bool IsExist<T>(System.Linq.Expressions.Expression<Func<T, bool>> predicate) where T : class

{

return dbContext.Set<T>().Any(predicate);

}

public void AddList<T>(IEnumerable<T> obj) where T : class

{

dbContext.Set<T>().AddRange(obj);

}

public int Save()

{

return dbContext.SaveChanges();

}

public int AddAndSave<T>(T model) where T : class

{

dbContext.Set<T>().Add(model);

return dbContext.SaveChanges();

}

public int UpdateAndSave<T>(T model) where T : class

{

dbContext.Entry(model).State = EntityState.Modified;

return dbContext.SaveChanges();

}

public int DeleteAndSave<T>(Guid id) where T : class, IRootModel, new()

{

dbContext.Entry(new T { Id = id }).State = EntityState.Deleted;

return dbContext.SaveChanges();

}

public IDbContextTransaction BeginTransaction()

{

return dbContext.Database.BeginTransaction();

}

public int ExecuteQuery(string query, params object[] parameters)

{

return dbContext.Database.ExecuteSqlRaw(query, parameters);

}

public IQueryable<T> FromQuery<T>(string sql, params object[] parameters) where T : class

{

return dbContext.SqlQuery<T>(sql, parameters);

}

public T GetObject<T>(string sql, params object[] parameters) where T : class

{

return dbContext.SqlQuery<T>(sql, parameters).FirstOrDefault();

}

}

public static class EFExtensions

{

public static IQueryable<T> IncludeNav<T, TProperty>(this IQueryable<T> query, Expression<Func<T, TProperty>> expression) where T : class

{

return query.Include(expression);

}

public static IQueryable<T> IncludeNav<T>(this IQueryable<T> query, params string[] navProperties) where T : class

{

foreach (var navProperty in navProperties)

query = query.Include(navProperty);

return query;

}

public static IQueryable<T> SqlQuery<T>(this DbContext db, string sql, params object[] parameters) where T : class

{

ContextForQueryType<T> db2 = new(db.Database.GetDbConnection());

return db2.Set<T>().FromSqlRaw(sql, parameters);

}

}

public class ContextForQueryType<T> : DbContext where T : class

{

private readonly DbConnection connection;

public ContextForQueryType(DbConnection connection)

{

this.connection = connection;

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(connection, options => options.EnableRetryOnFailure());

base.OnConfiguring(optionsBuilder);

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<T>().HasNoKey();

base.OnModelCreating(modelBuilder);

}

}

**10.2.3 Account Repository**

public class AccountRepository : Repository

{

readonly AppDbContext dbContext;

public AccountRepository(AppDbContext dbContext) : base(dbContext)

{

this.dbContext = dbContext;

}

private readonly string baseQuery = $"SELECT \* FROM Users ";

public IQueryable<DepartmentResponse> GetDepartments()

{

string query = @"select D.Id, DeptName, HODId, U.Name as HODName from Departments D

LEFT JOIN Employees E

ON D.HODId = E.Id

INNER JOIN Users U

ON U.Id = D.HODId

ORDER BY D.CreatedOn DESC ";

var result = FromQuery<DepartmentResponse>(query);

return result;

}

public IQueryable<UserResponse> GetUsers()

{

var result = FromQuery<UserResponse>(baseQuery + " ORDER BY Users.CreatedOn ");

return result;

}

public UserResponse GetUser(Guid id)

{

return GetObject<UserResponse>(baseQuery + $" WHERE U.Id = '{id}' ");

}

public IQueryable<ContactResponse> GetContacts()

{

string query = @" SELECT \* FROM CONTACTS";

return FromQuery<ContactResponse>(query);

}

public int DeleteContact(Guid id)

{

string query = $@" Delete from Contacts WHERE id ='{id}' ";

return ExecuteQuery(query);

}

public LoginResponse GetStudentUserDetails(Guid userId)

{

string query = $@"SELECT Students.SemesterId as SemesterId,

Courses.Id as CourseId,Departments.Id as DepartmentId FROM Users

Left JOIN Students

ON Users.Id = Students.Id

LEFT JOIN Semesters

ON Semesters.Id = Students.SemesterId

LEFT JOIN Courses

ON Semesters.CourseId = Courses.Id

Left JOIN Departments

ON Courses.DepartmentId = Departments.Id

Where Users.Id = '{userId}' ";

var result = GetObject<LoginResponse>(query);

return result;

}

public LoginResponse GetEmployeeUserDetails(Guid userId)

{

string query = $@"SELECT Departments.Id AS DepartmentId , SemesterId=NEWID(),

CourseId=NEWID() FROM Users

Left JOIN Employees

ON Users.Id = Employees.Id

Left JOIN Departments

ON Employees.DepartmentId = Departments.Id

WHERE Users.Id = '{userId}' ";

var result = GetObject<LoginResponse>(query);

return result;

}

public async Task<ProfileResponse> GetStudentProfileDetails(Guid userId)

{

string query = $@"select s.PhoneNo,s.RegNo from Users u

LEFT JOIN Students s

ON u.Id = s.Id

where u.Id ='{userId}' ";

var result = GetObject<ProfileResponse>(query);

return result;

}

public int UpdateStudentDetails(ProfileResponse profileResponse)

{

string query = @$"UPDATE Students

SET PhoneNo ='{profileResponse.PhoneNo}',RegNo ='{profileResponse.RegNo}'

where id ='{profileResponse.Id}' ";

var result = ExecuteQuery(query);

return result;

}

public StudentDetailsResponse GetStudentDetailsById(Guid id)

{

string query = @$"SELECT Students.Id,Users.Name,Users.UserName,Users.Email,Users.ImagePath,Users.CreatedOn,

Users.UserStatus,Students.PhoneNo,Students.RegNo,Students.Batch,

Departments.DeptName,Courses.CourseName,Semesters.Sem FROM Users

INNER JOIN Students

ON Users.Id=Students.Id

INNER JOIN Semesters

ON Students.SemesterId = Semesters.Id

INNER JOIN Courses

ON Semesters.CourseId = Courses.Id

INNER JOIN Departments

ON Courses.DepartmentId=Departments.Id

WHERE Students.Id ='{id}' ";

var result = GetObject<StudentDetailsResponse>(query);

return result;

}

public EmployeeDetailsResponse GetEmployeeDetailsById(Guid id)

{

string query = @$"select Users.Id,Users.ImagePath, Users.Name,Users.UserStatus, Users.UserName,Users.Email,Users.UserRole,Employees.PhoneNo,Departments.DeptName,Employees.EmpCode from Employees

INNER JOIN Users

ON Employees.Id= Users.Id

INNER JOIN Departments

ON Departments.Id=Employees.DepartmentId

where employees.id ='{id}' ";

var result = GetObject<EmployeeDetailsResponse>(query);

return result;

}

public int UpdateFacultyDetails(ProfileResponse profileResponse)

{

string query = @$"UPDATE Employees

SET PhoneNo ='{profileResponse.EmpPhoneNo}'

where id ='{profileResponse.Id}' ";

var result = ExecuteQuery(query);

return result;

}

}

**10.3 DbContext**

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)

{

ChangeTracker.QueryTrackingBehavior = QueryTrackingBehavior.NoTracking;

ChangeTracker.AutoDetectChangesEnabled = false;

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

var dbContextHandler = new DbContextHandler();

dbContextHandler.SetModelBuilderSettings(modelBuilder);

dbContextHandler.SeedInitialData(modelBuilder);

}

#region DbSet Properties

public DbSet<User> Users { get; set; }

public DbSet<Department> Departments { get; set; }

public DbSet<Course> Courses { get; set; }

public DbSet<Semester> Semesters { get; set; }

public DbSet<Employee> Employees { get; set; }

public DbSet<Student> Students { get; set; }

public DbSet<Topper> Toppers { get; set; }

public DbSet<Slider> Sliders { get; set; }

public DbSet<Contact> Contacts { get; set; }

public DbSet<Notification> Notifications { get; set; }

public DbSet<Subject> Subjects { get; set; }

public DbSet<Question> Questions { get; set; }

public DbSet<Paper> Papers { get; set; }

public DbSet<PaperQuestion> PaperQuestions { get; set; }

public DbSet<Result> Results { get; set; }

public DbSet<Score> Scores { get; set; }

public DbSet<FinalResult> FinalResults { get; set; }

#endregion DbSet Properties

}

**10.4**  **Account Manager**

public class AccountManager

{

private readonly AccountRepository \_repository;

private readonly IMapper \_mapper;

private readonly IMailService \_mailService;

public AccountManager(AccountRepository repository, IMapper mapper, IMailService mailService)

{

\_repository = repository;

\_mapper = mapper;

\_mailService = mailService;

}

public List<DepartmentResponse> GetDepartments()

{

return \_repository.GetDepartments().ToList();

}

#region AddUser

public int AddUser(SignupRequest request)

{

request.Id = Guid.NewGuid();

string uName = "";

request.Password = ("OEM-" + request.Name.Split(' ').FirstOrDefault()).ToLower();

string[] nameParts = request.Name.Split(' ');

if (nameParts.Length > 1)

uName = nameParts[0] + nameParts[1];

else uName = nameParts[0];

request.UserName = GetUniqueUserName(uName);

request.UserStatus = Status.Active;

var user = \_mapper.Map<SignupRequest, User>(request);

user.ImagePath = "default.png";

user.Salt = AppEncryption.CreateSalt();

user.Password = AppEncryption.CreatePasswordHash(user.Password, user.Salt);

if (\_repository.FindBy<User>(x => x.Email == request.Email).Any()) return -1;

if (\_repository.FindBy<Employee>(x => x.EmpCode == request.EmpCode).Any()) return -2;

var successValue = \_repository.AddAndSave(user);

if (request.UserRole == UserRole.Faculty || request.UserRole == UserRole.HOD)

{

Employee employee = new ();

employee.Id = request.Id;

employee.EmpCode = request.EmpCode;

employee.PhoneNo = "+91\*\*\*\*\*\*\*\*\*";

employee.DepartmentId = request.DepartmentId;

if(request.UserRole == UserRole.HOD)

{

var dep = \_repository.FindBy<Department>(x => x.Id == request.DepartmentId).FirstOrDefault();

if(dep.HODId != null)

{

var uzer = \_repository.FindBy<User>(x => x.Id == dep.HODId).FirstOrDefault();

var uuu = \_mapper.Map<User, User>(uzer);

uuu.UserRole = UserRole.Faculty;

\_repository.UpdateAndSave(uuu);

}

Department dept = new()

{

Id = dep.Id,

HODId = request.Id,

DeptName = dep.DeptName,

CreatedOn = dep.CreatedOn,

UpdatedOn = DateTime.Now,

};

\_repository.UpdateAndSave(dept);

}

employee.CreatedOn = DateTime.Now;

var reval = \_repository.AddAndSave(employee);

}

if (successValue >= 1)

{

var x = SendEmailToUser(request);

}

return successValue;

}

public List<DepartmentResponse> GetAllDepartments()

{

return \_repository.GetAll<Department>().Select(x => new DepartmentResponse

{

Id = x.Id,

DeptName = x.DeptName,

HODName = x.DeptName,

}).ToList();

}

public string GetUniqueUserName(string name)

{

string userName = name;

int incementValue = 1;

while (\_repository.FindBy<User>(x => x.UserName == userName).Any())

{

string newUsername = "";

newUsername = userName + incementValue.ToString();

userName = newUsername;

incementValue++;

}

return userName.ToLower();

}

public async Task SendEmailToUser(SignupRequest request)

{

MailRequestTo mailRequestTo = new MailRequestTo();

mailRequestTo.ToEmail = request.Email;

mailRequestTo.Subject = "Online Examination Credentials";

StringBuilder mailBody = new StringBuilder();

mailBody.AppendFormat("<h2>Dear " + request.Name.ToUpper() + "</h2><br/>");

mailBody.AppendFormat("<h5>Congratulations!You've successfully signed up for Online Examination System!</h5><br/>");

mailBody.AppendFormat("<h5>This Email includes your account details, so please keep it safe!</h5><br/>");

mailBody.AppendFormat("<h5>Alert! Must Change Your Cradentials Go To Edit Profile For Updation Your Details</h5><br/>");

mailBody.AppendFormat("<h3>Your User Name : " + (request.UserName).ToLower() + "</h3><br/>");

mailBody.AppendFormat("<h3>Your Password : " + request.Password + "</h3><br/>");

mailBody.AppendFormat("<h2>Thanks,</h2><br/>");

mailBody.AppendFormat("<h3>Online Examination Team!</h3><br/>");

mailRequestTo.Body = mailBody.ToString();

await \_mailService.SendEmailAsync(mailRequestTo);

}

#endregion

#region AddStudent

public int AddStudent(StudentRequest request)

{

request.Id = Guid.NewGuid();

string uName = "";

request.Password = ("OEM-" + request.Name.Split(' ').FirstOrDefault()).ToLower();

string[] nameParts = request.Name.Split(' ');

if (nameParts.Length > 1)

uName = nameParts[0] + nameParts[1];

else uName = nameParts[0];

request.UserName = GetUniqueUserName(uName);

request.UserStatus = Status.Active;

var user = \_mapper.Map<StudentRequest, User>(request);

user.UserRole = UserRole.Student;

user.ImagePath = "abc";

user.Salt = AppEncryption.CreateSalt();

user.Password = AppEncryption.CreatePasswordHash(user.Password, user.Salt);

if (request.SemesterId == Guid.Empty) return -3;

user.Student = new Student

{

Id = request.Id,

RegNo = request.RegNo,

SemesterId = request.SemesterId,

PhoneNo = "+91\*\*\*\*\*\*\*\*\*",

Batch = request.Batch,

CreatedOn = DateTime.Now,

};

if (\_repository.FindBy<User>(x => x.Email == request.Email).Any()) return -1;

if (\_repository.FindBy<Student>(x => x.RegNo == request.RegNo).Any()) return -2;

var successValue = \_repository.AddAndSave(user);

if (successValue >= 1)

{

var x = SendEmailToStudent(request);

}

return successValue;

}

public async Task SendEmailToStudent(StudentRequest request)

{

MailRequestTo mailRequestTo = new MailRequestTo();

mailRequestTo.ToEmail = request.Email;

mailRequestTo.Subject = "Online Exam Mart Credentials";

StringBuilder mailBody = new StringBuilder();

mailBody.AppendFormat("<h2>Dear " + request.Name.ToUpper() + "</h2><br/>");

mailBody.AppendFormat("<h5>Congratulations!You've successfully signed up for Online Exam Mart!</h5><br/>");

mailBody.AppendFormat("<h5>This Email includes your account details, so please keep it safe!</h5><br/>");

mailBody.AppendFormat("<h5>Alert! Must Change Your Cradentials Go To Edit Profile For Updation Your Details</h5><br/>");

mailBody.AppendFormat("<h3>Your User Name : " + (request.UserName).ToLower() + "</h3><br/>");

mailBody.AppendFormat("<h3>Your Password : " + request.Password + "</h3><br/>");

mailBody.AppendFormat("<h2>Thanks,</h2><br/>");

mailBody.AppendFormat("<h3>Online Exam Mart Team!</h3><br/>");

mailRequestTo.Body = mailBody.ToString();

await \_mailService.SendEmailAsync(mailRequestTo);

}

#endregion

#region Login

public ResponseModel<UserResponse> UserLogin(Login login)

{

var responseModel = new ResponseModel<UserResponse>();

User? user = \_repository.FindBy<User>(x => x.UserName == login.UserName).FirstOrDefault();

if (user == null || !AppEncryption.ComparePassword(user.Password, login.Password, user.Salt))

{

responseModel.HasError = true;

responseModel.Message = "Invalid Credentials";

return responseModel;

}

UserResponse userResponsee = new UserResponse();

userResponsee.Id = user.Id;

userResponsee.Name = user.Name;

userResponsee.UserName = user.UserName;

userResponsee.Email = user.Email;

userResponsee.ImagePath = user.ImagePath;

userResponsee.UserStatus = user.UserStatus;

userResponsee.UserRole = user.UserRole;

if (user.UserRole == UserRole.Student)

{

var loginResponse = \_repository.GetStudentUserDetails(user.Id);

userResponsee.SemesterId = loginResponse.SemesterId;

userResponsee.CourseId = loginResponse.CourseId;

userResponsee.DepartmentId = loginResponse.DepartmentId;

}

else if (user.UserRole == UserRole.Faculty || user.UserRole == UserRole.HOD)

{

var loginResponse = \_repository.GetEmployeeUserDetails(user.Id);

userResponsee.DepartmentId = loginResponse.DepartmentId;

}

responseModel.Result = userResponsee;

return responseModel;

}

#endregion

public string ForgetPassword(ForgotEmail email, string link, [FromServices] IEmailService emailService)

{

User? user = \_repository.FindBy<User>(x => x.Email == email.Email).FirstOrDefault();

if (user != null)

{

Guid guid = Guid.NewGuid();

user.ResetCode = guid.ToString();

link += guid;

var value = \_repository.UpdateAndSave(user);

if (value >= 1)

{

var x = SendResetEmail(user, link);

return "success";

}

}

return "NotExist";

}

public async Task SendResetEmail(User request, string link)

{

MailRequestTo mailRequestTo = new MailRequestTo();

mailRequestTo.ToEmail = request.Email;

mailRequestTo.Subject = "Reset Password (Online Examination Mart)";

StringBuilder mailBody = new StringBuilder();

mailBody.AppendFormat("Please Click on the following link to reset your password <br>");

mailBody.AppendFormat($"<a href=\"{link}\">{link}</a>");

mailRequestTo.Body = mailBody.ToString();

await \_mailService.SendEmailAsync(mailRequestTo);

}

public User ResetPassword(ResetPassword resetPassword)

{

User user = \_repository.FindBy<User>(x => x.ResetCode == resetPassword.GUID).FirstOrDefault();

if (user != null)

{

user.Salt = AppEncryption.CreateSalt();

user.Password = AppEncryption.CreatePasswordHash(resetPassword.NewPassword, user.Salt);

user.ResetCode = null;

\_repository.UpdateAndSave(user);

return user;

}

return null;

}

public StudentDetailsResponse GetStudentDetailsById(Guid id)

{

return \_repository.GetStudentDetailsById(id);

}

public EmployeeDetailsResponse GetEmployeeDetailsById(Guid id)

{

return \_repository.GetEmployeeDetailsById(id);

}

public int ChangePassword(ChangePassword changePassword)

{

User userAccount = \_repository.GetById<User>(changePassword.Id);

if (userAccount != null)

{

string dbSalt = userAccount.Salt;

string dbPassword = userAccount.Password;

string oldPass = AppEncryption.CreatePasswordHash(changePassword.OldPassword, dbSalt);

if (oldPass.Equals(dbPassword))

{

userAccount.Salt = AppEncryption.CreateSalt();

userAccount.Password = AppEncryption.CreatePasswordHash(changePassword.NewPassword, userAccount.Salt);

return \_repository.UpdateAndSave(userAccount);

}

else

{

return 0;

}

}

else

{

return -3;

}

}

public Task<ProfileResponse> GetUserById(Guid userId, string userRole)

{

return Task.Run(() => \_repository.FindBy<User>(x => x.Id == userId).IncludeNav(a => a.Student).IncludeNav(b => b.Employee).Select(x => new ProfileResponse

{

Id = x.Id,

UserName = x.UserName,

Name = x.Name,

Email = x.Email,

ImagePath = x.ImagePath,

UserRole = x.UserRole,

PhoneNo = x.Student.PhoneNo,

RegNo = x.Student.RegNo,

EmpPhoneNo=x.Employee.PhoneNo

}).FirstOrDefault());

}

public int UpdateUser(ProfileResponse profileResponse, string webRootPath, string userRole)

{

var user = \_repository.GetById<User>(profileResponse.Id);

user.Name = profileResponse.Name;

if (\_repository.FindBy<User>(x => x.Id == profileResponse.Id && x.Email == profileResponse.Email).Any())

{

user.Email = profileResponse.Email;

}

else if (\_repository.FindBy<User>(x => x.Email == profileResponse.Email).Any()) return -1;

else user.Email = profileResponse.Email;

if (profileResponse.File != null)

{

if (user.ImagePath != webRootPath)

{

FileManager.DeleteFile(webRootPath, user.ImagePath);

}

user.ImagePath = FileManager.SaveFile(profileResponse.File, webRootPath);

}

if (userRole == "Student")

{

var std = \_repository.UpdateStudentDetails(profileResponse);

}

if (userRole == "Faculty")

{

var std = \_repository.UpdateFacultyDetails(profileResponse);

}

return \_repository.UpdateAndSave(user);

}

public List<UserResponse> GetUsers()

{

return \_repository.FindBy<User>(x => x.UserRole == UserRole.Admin).OrderByDescending(y => y.CreatedOn).Select(z => new UserResponse

{

Id = z.Id,

UserName = z.UserName,

Name = z.Name,

UserRole = z.UserRole,

UserStatus = z.UserStatus,

Email = z.Email,

CreatedOn = z.CreatedOn,

ImagePath = z.ImagePath,

}).ToList(); ;

}

public List<UserResponse> GetEmployees()

{

return \_repository.FindBy<User>(x => x.UserRole == UserRole.Faculty || x.UserRole== UserRole.HOD).OrderByDescending(y => y.CreatedOn).Select(z => new UserResponse

{

Id = z.Id,

UserName = z.UserName,

Name = z.Name,

UserRole = z.UserRole,

UserStatus = z.UserStatus,

Email = z.Email,

CreatedOn = z.CreatedOn

}).ToList(); ;

}

public List<UserResponse> GetStudents()

{

return \_repository.FindBy<User>(x => x.UserRole == UserRole.Student).OrderByDescending(y => y.CreatedOn).Select(z => new UserResponse

{

Id = z.Id,

UserName = z.UserName,

Name = z.Name,

UserRole = z.UserRole,

UserStatus = z.UserStatus,

Email = z.Email,

CreatedOn = z.CreatedOn

}).ToList(); ;

}

public int AddContact(ContactRequest contactRequest)

{

var returnValue = \_mapper.Map<ContactRequest, Contact>(contactRequest);

return \_repository.AddAndSave<Contact>(returnValue);

}

public IQueryable<ContactResponse> GetContacts()

{

return \_repository.GetContacts();

}

public int DeleteContact(Guid id)

{

return \_repository.DeleteContact(id);

}

public bool ChangeUserStatus(Guid id)

{

var x = \_repository.GetById<User>(id);

if (x.UserStatus == Status.Active)

{

x.UserStatus = Status.Inactive;

\_repository.UpdateAndSave(x);

return false;

}

else

{

x.UserStatus = Status.Active;

\_repository.UpdateAndSave(x);

return true;

}

}

}

**10.5 WebApp Controllers**

**10.5.1 Account Controller**

public class AccountController : Controller

{

private readonly AccountManager \_accountManager;

private readonly IMapper \_mapper;

private readonly IMailService \_mailService;

private readonly IWebHostEnvironment environment;

private int returnValue;

public AccountController(AccountRepository repository, IMapper mapper, IMailService mailService, IWebHostEnvironment environment)

{

\_accountManager = new AccountManager(repository, mapper, mailService);

\_mapper = mapper;

\_mailService = mailService;

this.environment = environment;

}

[Route("index")]

public IActionResult Index()

{

return View();

}

[Route("signup")]

[HttpGet]

public IActionResult Signup()

{

ViewBag.Departments = \_accountManager.GetAllDepartments();//change here

return View();

}

[Route("signup")]

[HttpPost]

public IActionResult Signup(SignupRequest request)

{

var returnValue = \_accountManager.AddUser(request);

ViewBag.returnValue = returnValue;

ViewBag.Departments = \_accountManager.GetDepartments();

return View();

}

[Route("addstudent")]

[HttpGet]

public IActionResult AddStudent()

{

ViewBag.Departments = \_accountManager.GetDepartments();

return View();

}

[Route("addstudent")]

[HttpPost]

public IActionResult AddStudent(StudentRequest studentRequest)

{

//if (ModelState.IsValid)

//{

var returnValue = \_accountManager.AddStudent(studentRequest);

ViewBag.returnValue = returnValue;

ViewBag.Departments = \_accountManager.GetDepartments();

//}

return View();

}

[AllowAnonymous]

[Route("login")]

[HttpGet]

public IActionResult Login(string? returnUrl)

{

return View();

}

[AllowAnonymous]

[Route("login")]

[HttpPost]

public async Task<IActionResult> Login(Login login, string? returnUrl)

{

if (ModelState.IsValid)

{

var response = \_accountManager.UserLogin(login);

if (response.HasError)

{

ViewBag.Error = "Invalid";

return View();

}

var user = response.Result;

if (user.UserStatus == Status.Inactive)

{

ViewBag.Error = "Blocked";

return View();

}

var identity = new ClaimsIdentity(CookieAuthenticationDefaults.AuthenticationScheme);

identity.AddClaim(new Claim(AppClaimTypes.UserId, user.Id.ToString()));

identity.AddClaim(new Claim(AppClaimTypes.Name, user.Name));

identity.AddClaim(new Claim(AppClaimTypes.Email, user.Email));

identity.AddClaim(new Claim(AppClaimTypes.Role, user.UserRole.ToString()));

identity.AddClaim(new Claim(AppClaimTypes.UserName, user.UserName));

identity.AddClaim(new Claim(AppClaimTypes.ImagePath, user.ImagePath.ToString()));

if (user.UserRole == UserRole.Student)

{

identity.AddClaim(new Claim(AppClaimTypes.SemesterId, user.SemesterId.ToString()));

identity.AddClaim(new Claim(AppClaimTypes.CourseId, user.CourseId.ToString()));

identity.AddClaim(new Claim(AppClaimTypes.DepartmentId, user.DepartmentId.ToString()));

}

else if (user.UserRole == UserRole.Faculty || user.UserRole == UserRole.HOD)

{

identity.AddClaim(new Claim(AppClaimTypes.DepartmentId, user.DepartmentId.ToString()));

}

var principal = new ClaimsPrincipal(identity);

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal, new AuthenticationProperties()

{

IsPersistent = login.RememberMe,

AllowRefresh = true,

IssuedUtc = DateTime.Now,

ExpiresUtc = DateTime.Now.AddDays(10)

});

if (returnUrl != null && Url.IsLocalUrl(returnUrl))

return Redirect(returnUrl);

else if (user.UserRole == UserRole.Admin)

return RedirectToAction("Index", "Dashboard");

else if (user.UserRole == UserRole.Faculty || user.UserRole == UserRole.HOD)

return RedirectToAction("Index", "Employee");

else if (user.UserRole == UserRole.Student)

return RedirectToAction("Index", "Student");

}

else

{

ViewBag.Message = "User not found";

}

return View();

}

[Route("forgotpassword")]

[AllowAnonymous]

public IActionResult ForgotPassword()

{

return View();

}

[Route("forgotpassword")]

[AllowAnonymous]

[HttpPost]

public ActionResult ForgotPassword(ForgotEmail forgotemail, [FromServices] SharedLibrary.IEmailService emailService)

{

string link = Request.GetEncodedUrl().Replace(Request.Path.ToUriComponent(), "/Account/ResetPassword?guid=");

string msg = \_accountManager.ForgetPassword(forgotemail, link, emailService);

if (msg == "success")

ViewBag.Message = "Check your Registered Email to Reset your Password";

else

ViewBag.Message = "Invalid Email";

return View();

}

[AllowAnonymous]

[HttpGet]

[Route("resetpassword")]

public ActionResult ResetPassword()

{

ViewBag.guid = HttpContext.Request.Query["guid"];

return View();

}

[AllowAnonymous]

[HttpPost]

[Route("resetpassword")]

public async Task<ActionResult> ResetPassword(ResetPassword resetPassword)

{

var user = \_accountManager.ResetPassword(resetPassword);

if (user == null)

{

ViewBag.Message = "Error";

return View();

}

else

{

if (resetPassword.IsChecked)

{

var identity = new ClaimsIdentity(CookieAuthenticationDefaults.AuthenticationScheme);

identity.AddClaim(new Claim(AppClaimTypes.UserId, user.Id.ToString()));

identity.AddClaim(new Claim(AppClaimTypes.Email, user.Email));

identity.AddClaim(new Claim(AppClaimTypes.Name, user.Name));

var principal = new ClaimsPrincipal(identity);

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal, new AuthenticationProperties()

{

IsPersistent = true,

AllowRefresh = true,

IssuedUtc = DateTime.UtcNow,

ExpiresUtc = DateTime.UtcNow.AddHours(10)

});

if (user.UserRole == UserRole.Admin)

return RedirectToAction("Index", "Account");

else if (user.UserRole == UserRole.Faculty)

return RedirectToAction("Index", "Faculty");

else if (user.UserRole == UserRole.Student)

return RedirectToAction("Index", "Student");

}

ViewBag.Message = "Success";

}

return View();

}

[AllowAnonymous]

[HttpGet("changepassword")]

public IActionResult ChangePassword()

{

return View();

}

[AllowAnonymous]

[Route("changepassword")]

[HttpPost]

public IActionResult ChangePassword(ChangePassword changePassword)

{

changePassword.Id = User.GetUserId();

ViewBag.returnValue = \_accountManager.ChangePassword(changePassword);

return View();

}

[Route("logout")]

[AllowAnonymous]

public async Task<IActionResult> Logout()

{

await HttpContext.SignOutAsync();

TempData.Remove("UpdatedImage");

return Redirect("/");

}

[Route("profile")]

[AllowAnonymous]

[HttpGet]

public async Task<ActionResult> Profile()

{

ViewBag.returnValue = returnValue;

var user = await \_accountManager.GetUserById(User.GetUserId(), User.GetRole());

if (user.ImagePath != User.GetImage())

{

TempData["UpdatedImage"] = user.ImagePath;

}

return View(user);

}

[Route("editprofile")]

[AllowAnonymous]

[HttpPost]

public ActionResult EditProfile(ProfileResponse profileResponse)

{

var role = User.GetRole();

returnValue = \_accountManager.UpdateUser(profileResponse, environment.WebRootPath, role);

return RedirectToAction(nameof(Profile), returnValue);

}

[Route("users")]

public IActionResult Users()

{

var userResponses = \_accountManager.GetUsers();

return View(userResponses);

}

[HttpPost("ChangeUserStatus/{id}")]

public JsonResult ChangeUserStatus(Guid id)

{

var status = \_accountManager.ChangeUserStatus(id);

return Json(status);

}

[Route("employees")]

public IActionResult Employees()

{

var employees = \_accountManager.GetEmployees();

return View(employees);

}

[Route("employeedetails")]

public IActionResult EmployeeDetailss(Guid id)

{

var details = \_accountManager.GetEmployeeDetailsById(id);

return View(details);

}

[Route("students")]

public IActionResult Students()

{

var students = \_accountManager.GetStudents();

return View(students);

}

public IActionResult StudentDetails(Guid id) // used to show details to Admin

{

var details = \_accountManager.GetStudentDetailsById(id);

return View(details);

}

[Route("getcontacts")]

[HttpGet]

public IActionResult GetContacts()

{

var result = \_accountManager.GetContacts();

return View(result);

}

[Route("deletecontact")]

[HttpGet]

public IActionResult DeleteContact(Guid id)

{

var result = \_accountManager.DeleteContact(id);

return RedirectToAction(nameof(GetContacts));

}

}

**10.5.2 Paper Controller**

public class PaperController : Controller

{

private readonly PaperManger paperManager;

private readonly IMapper mapper;

private readonly IMailService mailService;

public PaperController(PaperRepository repository, IMapper mapper, IMailService mailService)

{

this.paperManager = new PaperManger(repository, mapper, mailService);

this.mapper = mapper;

this.mailService = mailService;

}

[HttpGet("create")]

public IActionResult Index()

{

ViewBag.Departments = paperManager.GetDepartments();

ViewBag.Batch = paperManager.GetBatchs();

return View();

}

[HttpGet("getpapers")]

public IActionResult GetPapers()

{

ViewBag.Departments = paperManager.GetDepartments();

return View();

}

[HttpGet("getpaper/{id}")]

public PartialViewResult GetPapersById(Guid id)

{

var papers = paperManager.GetPapersBySubjectId(id);

return PartialView("\_PaperListPartial", papers);

}

[HttpPost("postpaper")]

public JsonResult CreatePaper(PaperRequest paperRequest)

{

var returnValue = paperManager.AddPaper(paperRequest, User.GetRole());

return Json(returnValue);

}

[HttpGet("updatepaper")]

public IActionResult UpdatePaper(Guid Id)

{

var paper = paperManager.GetPaperById(Id);

return View(paper);

}

[HttpPost("updatepaper")]

public IActionResult UpdatePaper(PaperRequest paperRequest)

{

var returnValue = paperManager.UpdatePaper(paperRequest);

return RedirectToAction(nameof(GetPapers));

}

[HttpGet("addquestion/{subjectId?}/{Id?}")]

public IActionResult AddQuestion(Guid SubjectId, Guid Id,int maxMarks)

{

if (maxMarks > 0)

{

Response.Cookies.Append("maxMarks", maxMarks.ToString(), new CookieOptions { Expires = DateTime.Now.AddHours(8) });

}

ViewBag.Departments = paperManager.GetDepartments();

var questions = paperManager.GetQuestionsBySubId(SubjectId).ToList();

ViewBag.PaperId= Id;

return View(questions);

}

[HttpPost("addquestion")]

public JsonResult AddQuestion([FromBody]List<PaperQuestionRequest> paperQuestions)

{

var js = "error";

//int total = 0;

//for (int i = 0; i < paperQuestions.Count(); i++)

//{

// total = paperQuestions[i].Marks + paperQuestions[++i].Marks;

//}

var maxMarks = int.Parse(Request.Cookies["maxMarks"]);

var maxQuestionMarksSelected = paperQuestions.Sum(x => x.Marks);

if (maxQuestionMarksSelected <= maxMarks)

{

var returnValue = paperManager.AddQuestion(paperQuestions);

return Json(returnValue);

}

else

{

return Json(js);

}

}

[HttpGet("paperdetails/{id?}")]

public IActionResult PaperDetails(Guid id)

{

var paper = paperManager.GetPaperDetails(id);

return View(paper);

}

}

**10.5.3 Question Controller**

public class QuestionController : Controller

{

private readonly QuestionManager questionManager;

private readonly IMapper \_mapper;

public QuestionController(QuestionRepository repository, IMapper mapper)

{

questionManager = new QuestionManager(repository, mapper);

\_mapper = mapper;

}

[Route("index")]

[AllowAnonymous]

public IActionResult Index()

{

ViewBag.Departments = questionManager.GetDepartments();

// var questions = questionManager.GetQuestions();

return View();

}

[HttpPost("question")]

public PartialViewResult CreateQuestion(QuestionRequest questionrequest)

{

var res= questionManager.AddQuestion(questionrequest);

var questions = questionManager.GetQuestionsBySubId(questionrequest.SubjectId);

return PartialView("\_QuestionListPartial", questions);

}

[HttpGet("getquestionbysub/{Id?}")]

public PartialViewResult GetQuestionsBySub(Guid Id)

{

var questions = questionManager.GetQuestionsBySubId(Id);

return PartialView("\_QuestionListPartial", questions);

}

[HttpPost("UpdateQuestion")]

public IActionResult UpdateQuestion(QuestionRequest questionRequest)

{

var res= questionManager.UpdateQuestion(questionRequest);

return RedirectToAction(nameof(Index));

}

[HttpGet("questiondetails")]

public IActionResult QuestionDetails(Guid questionPaperId)

{

return View(questionManager.GetQuestionDetails(questionPaperId));

}

}

**10.6 Design**

**10.6.1**  **Layout**

body>

<div class="topnav" id="myTopnav">

<**a** **asp-action**="Index" **asp-controller**="Home"

style="color:aqua;text-decoration:none; font-size: 38px;font-family: 'Dancing Script', cursive;">Online Examination System</**a**>

<**a** **asp-action**="ContactUs" **asp-controller**="Home" style="font-size: 22px;" class="nav-links btn myTopnav"><i class="fa fa fa-phone icons" style="display:block"></i>Contact</**a**>

<**a** **asp-action**="AboutUs" **asp-controller**="Home" style="font-size: 22px;" class="nav-links btn myTopnav"><i class="fa fa-info-circle icons" style="display:block"></i>About</**a**>

@if (User.Identity.IsAuthenticated)

{

<style>

#ovii {

display: none;

}

</style>

}

<**a** id="ovii" **asp-action**="Login" **asp-controller**="Account" style="font-size: 22px;font-family: 'Fira Sans', sans-serif;" class="nav-links btn myTopnav"><i class="fa fa-lock icons" style="display:block"></i>Login</**a**>

@if (User.Identity.IsAuthenticated && User.IsInRole(UserRole.Admin.ToString()))

{

<**a** **asp-action**="Index" **asp-controller**="Dashboard" style="font-size: 20px;" class="nav-links btn myTopnav"><i class="fa fa-unlock icons" style="display:block"></i>Goto @User.GetRole() Panel</**a**>

}

else if (User.IsInRole(UserRole.Student.ToString()))

{

<**a** **asp-action**="Index" **asp-controller**="Student" style="font-size: 20px;" class="nav-links btn myTopnav"><i class="fa fa-unlock icons" style="display:block"></i>Goto @User.GetRole() Panel</**a**>

}

else if (User.IsInRole(UserRole.Faculty.ToString()) || User.IsInRole(UserRole.HOD.ToString()))

{

<**a** **asp-action**="Index" **asp-controller**="Employee" style="font-size: 20px;" class="nav-links btn myTopnav"><i class="fa fa-unlock icons" style="display:block"></i>Goto @User.GetRole() Panel</**a**>

}

<a href="javascript:void(0);" style="font-size:15px;" class="icon" onclick="myFunction()">&#9776;</a>

</div>

<div>

<main role="main">

@RenderBody()

</main>

</div>

<footer style="margin-top:-3%;">

<div class="footer-content">

<h3><**a** style="color:aqua;font-weight:bolder;font-family:'Dancing Script', cursive;" **asp-action**="Index" **asp-controller**="Home">OES</**a**></h3>

<center>The entire life is exam that never ends until the life itself ends.</center>

<ul class="socials">

<li><a href="#"><i class="fa fa-facebook"></i></a></li>

<li><a href="#"><i class="fa fa-twitter"></i></a></li>

<li><a href="#"><i class="fa fa-google-plus"></i></a></li>

<li><a href="#"><i class="fa fa-youtube"></i></a></li>

<li><a href="#"><i class="fa fa-linkedin-square"></i></a></li>

</ul>

<div class="row">

<div class="col-12">

<ul>

<h4 style="color:white ; font-weight:lighter">INFORMATION</h4>

<hr />

<h6 style="color:whitesmoke">Hazratbal Srinagar</h6>

<a href="https://www.kashmiruniversity.net/" target="\_blank">Kashmir University</a>

<h6 style="color:whitesmoke">+918825084050</h6>

</ul>

</div>

</div>

<div class="footer-bottom">

<p>copyright &copy;2021 <**a** style="font-family:'Dancing Script', cursive;font-weight:bolder;color:aqua;" **asp-controller**="Home" **asp-action**="index">ONLINE EXAMINATION SYSTEM</**a**> </p>

<div class="footer-menu">

<ul class="f-menu">

<li><**a** **asp-action**="AboutUs" **asp-controller**="Home">About</**a**></li>

<li><**a** **asp-controller**="Home" **asp-action**="ContactUs">Contact</**a**></li>

</ul>

</div>

</div>

</div>

**10.6.2 Enter Exam**

<style>

.time {

display: flex;

justify-content: end;

font-size: 25px;

font-weight: bolder;

}

.marks {

display: flex-block;

justify-content: end;

font-size: 25px;

font-weight: bolder;

margin-left: 16px;

margin-top: -27px;

}

</style>

@{

}

<!DOCTYPE HTML>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width initial-scale=1">

<meta name="description" content="This is a Exam designed for basic HTML, CSS, and JavaScript basics.">

<title>Exam</title>

<link href="https://fonts.googleapis.com/css?family=Share+Tech+Mono&display=swap" rel="stylesheet">

<**link** rel="stylesheet" type="text/css" href="~/css/examlayout.css">

<**link** rel="stylesheet" type="text/css" href="~/css/style.css">

</head>

<body>

<header>

<section class="title">

<h1>Exam<span class="sub-title">@Model?.SubjectName</span></h1>

</section>

<button type="button" class="hide hint-btn">Hint</button>

</header>

@if (ViewBag.Error != "error")

{

<div class="text-center time" id="count">

</div>

}

<div class="marks">

<div>

Min Marks : @Model.MinimumMarks

</div>

<div>

Max Marks : @Model.MaximumMarks

</div>

</div>

<main>

<article class="begin question-answer-wrapper">

<section class="hide results-wrapper">

<p class="quiz-end-score"></p>

<div class="quiz-end-feedback">

<p class="quiz-end-feedback-p"></p>

<div class="quiz-end-categories"></div>

</div>

<button type="button" class="retry-btn pass">Retry</button>

</section>

@{

var resultId = ViewBag.ResultId;

}

@if (ViewBag.Error == "error")

{

<div class="alert alert-danger">Paper already submitted</div>

}

else

{

<**form** **asp-action**="submitquestion" **asp-controller**="Student" id="postQuestion">

<input type="hidden" value="@resultId" name="resultId" />

<input type="hidden" value="@Model?.Id" name="paperId" />

<div id="questionDiv">

<**partial** **name**="\_QustionPaperPartial" **model**="@Model?.NextQuestion" />

</div>

<input type="submit" class="submit-btn submit-btn-portrait" id="submitExam" value="Submit" disabled>

</**form**>

}

</article>

</main>

<footer>

<section class="hide progress">

<p class="progress-fill">

Progress: <span class="progress-count">1 / @Model?.TotalQuestion</span>

</p>

</section>

</footer>

<div id="output">

</div>

<**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>

<**script** src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></**script**>

<**script** src="~/lib/jquery-validation/dist/jquery.validate.js"></**script**>

<**script** src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></**script**>

<**script** type="module" **src**="~/client/services/admin/studentService.js" **asp-append-version**="true"></**script**>

<script>

var examTime = @Model.ExamDateTime.Minute;

var examDuration = @Model.ExamDuration;

var currentTime = new Date();

var mins = currentTime.getMinutes();

var time = mins - examTime;

var totalResult = examDuration - time;

let secondss=59;

var count = totalResult - 1 ; //15 minute

document.getElementById('count').innerHTML=`${count}:${secondss} minutes to go..`;

console.log(count,secondss);

var intrVal= setInterval(function(){

secondss--;

document.getElementById('count').innerHTML=`${count}:${secondss} minutes to go..`;

if(count === 0 && secondss === 0){

clearInterval(intrVal);

clearInterval(interval);

console.log(count,secondss);

document.getElementById('count').innerHTML='Time Up!';

// or...

alert("You're out of time!");

document.getElementById('questionDiv').innerHTML="";

document.getElementById('questionDiv').innerHTML="Exam is Over";

document.getElementById('questionDiv').style="font-size:50px; font-weight:bolder; ";

document.getElementById('submitExam').style.display="none";

}

if(secondss === 0){

secondss = 60;

}

},1000);

var interval = setInterval(function(){

count--;

}, 60000);

</script>

</body>

</html>

Chapter 13

**FUTURE SCOPE**

# **Future Scope**

* As internet are smarter and flexible day by day. In future we will try to make our website more flexible and user friendly.
* We will add more features and will remove the bugs that will affect the functionality of the website and keep it up to date.
* We will make it more secure and fast as well.
* In future we will develop its Android and IOS app, so that it can reach to maximum people.
* We will add more products and delivery quicker than before. We will make payment method more flexible.

Chapter 13

**CONCLUSION**

# **CONCLUSION**

The project entitled ONLINE EXAMINATION SYSTEM was completed successfully. The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. The purpose of this project was to develop a website that will reduce the burden of Examiners and Examinees as much as possible. Moreover, this web application will be cost effective and will come with easy to use interface.

This project helped us in gaining valuable information and practical knowledge on several topics like designing web pages using HTML & CSS, usage of responsive templates and management of database using SQL SERVER 2019. The entire system is secured. Also, the project helped us understanding about the development phases of a project and software development life cycle. We learned how to test different features of a project.

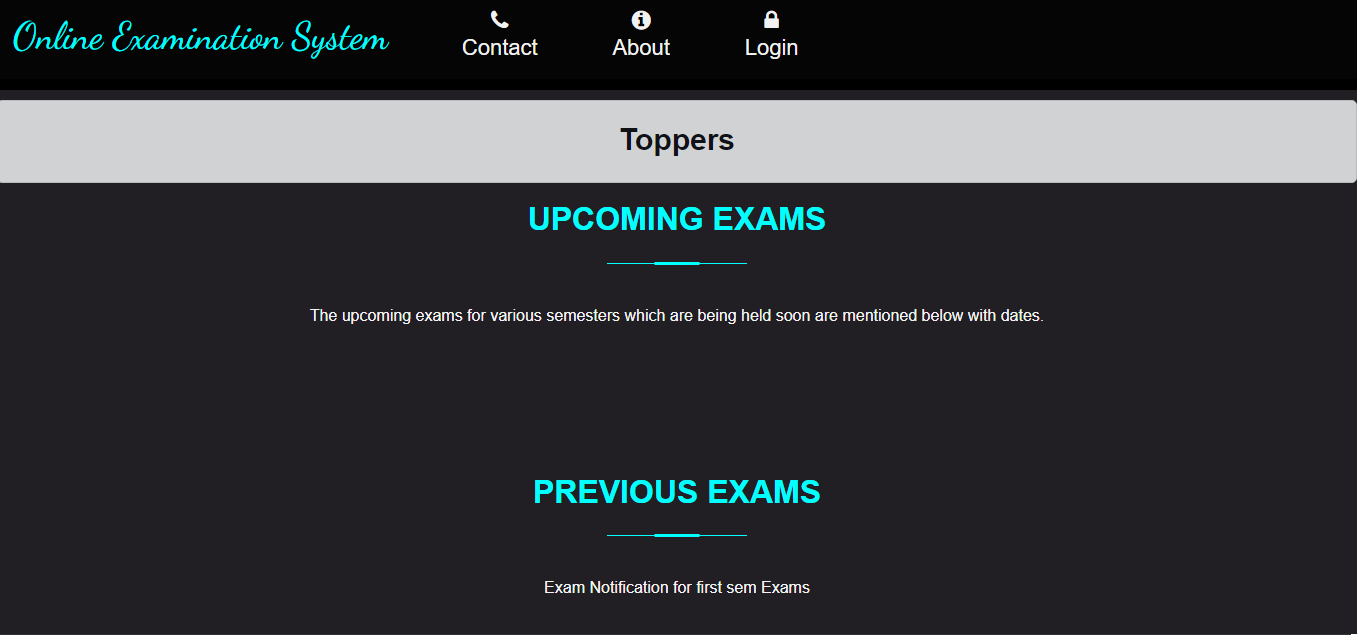
This project has given us great satisfaction in having designed a website that will take exams online and generate results on the go, consuming less time and effort simultaneously.

There is a scope for further development in our project to a great extent. A number of features can be added to this system in future like making IOS and Android Applications so that it will be more user friendly and easy to use. It will be made more secure and flexible and we will add more features to this project in future. It will be up to date by adding new features and other necessary things as well.

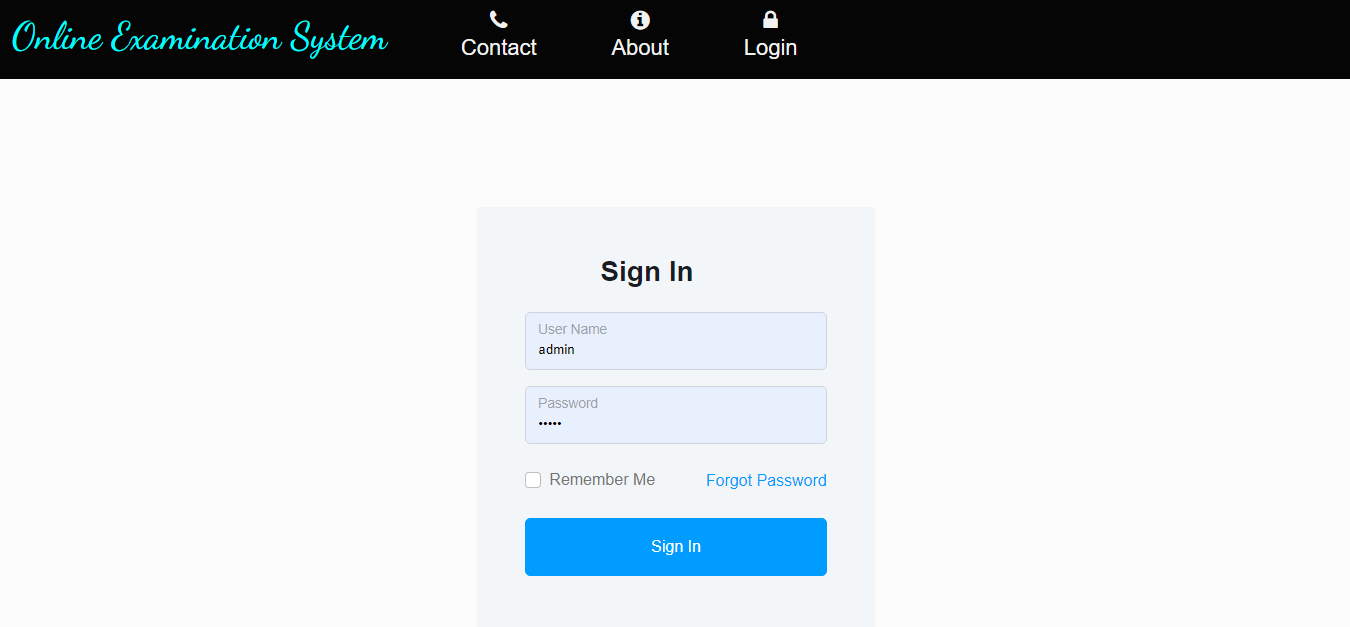
Chapter 13

**SCREENSHOTS**

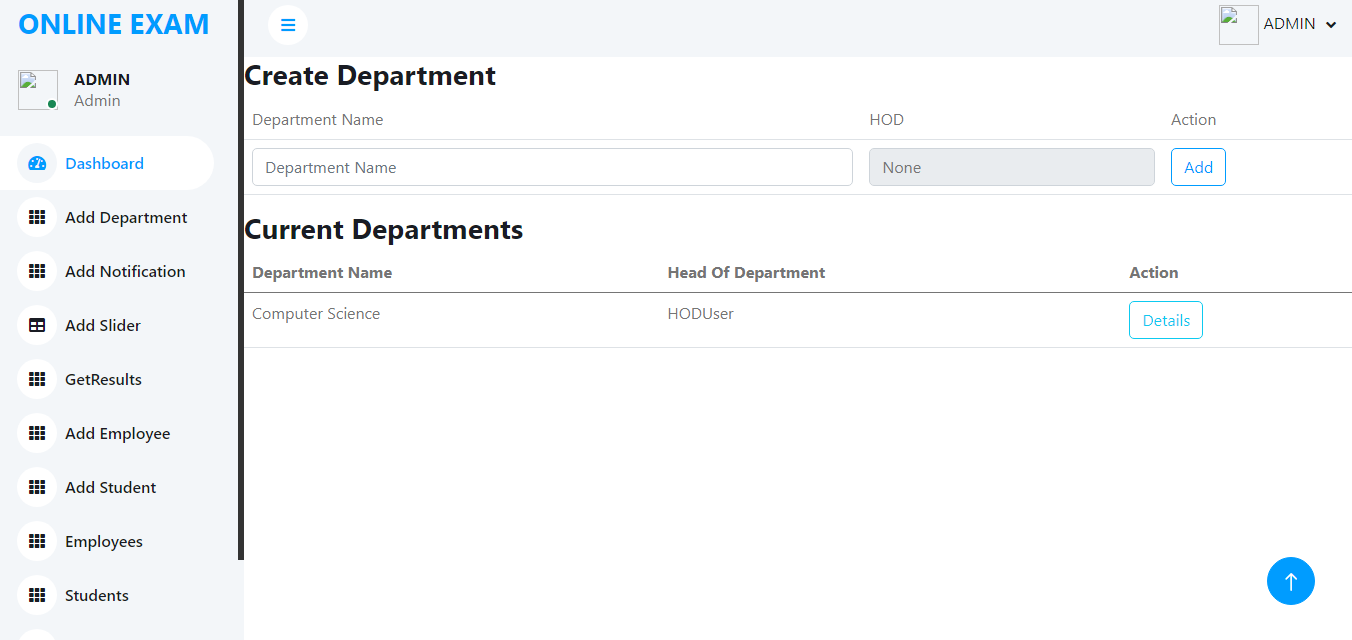
**13.1 Home Page**

****

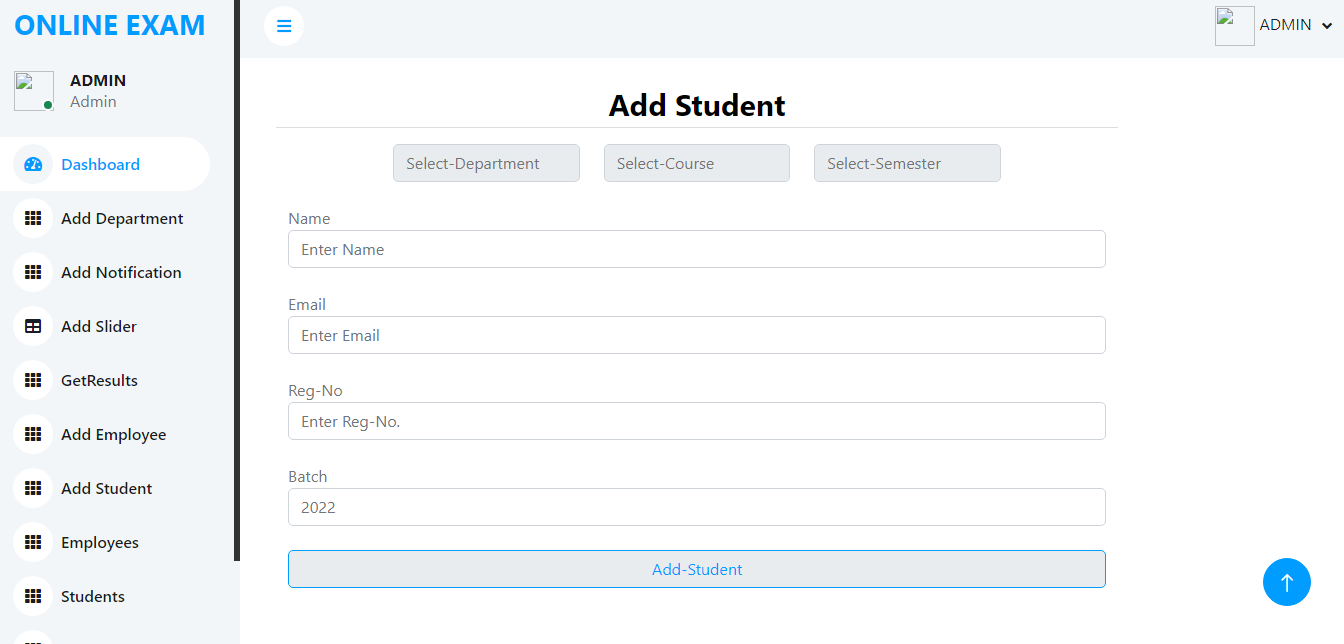
**13.2 Login Page**

****

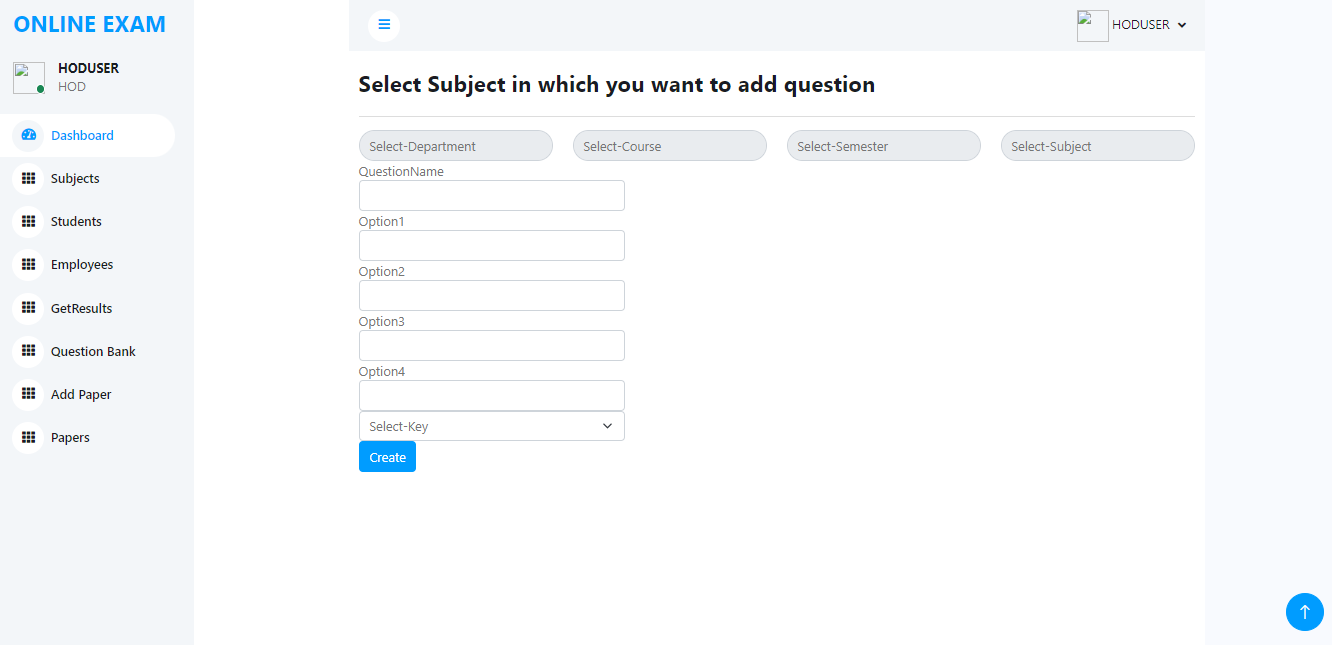
**13.3 Add Department**

****

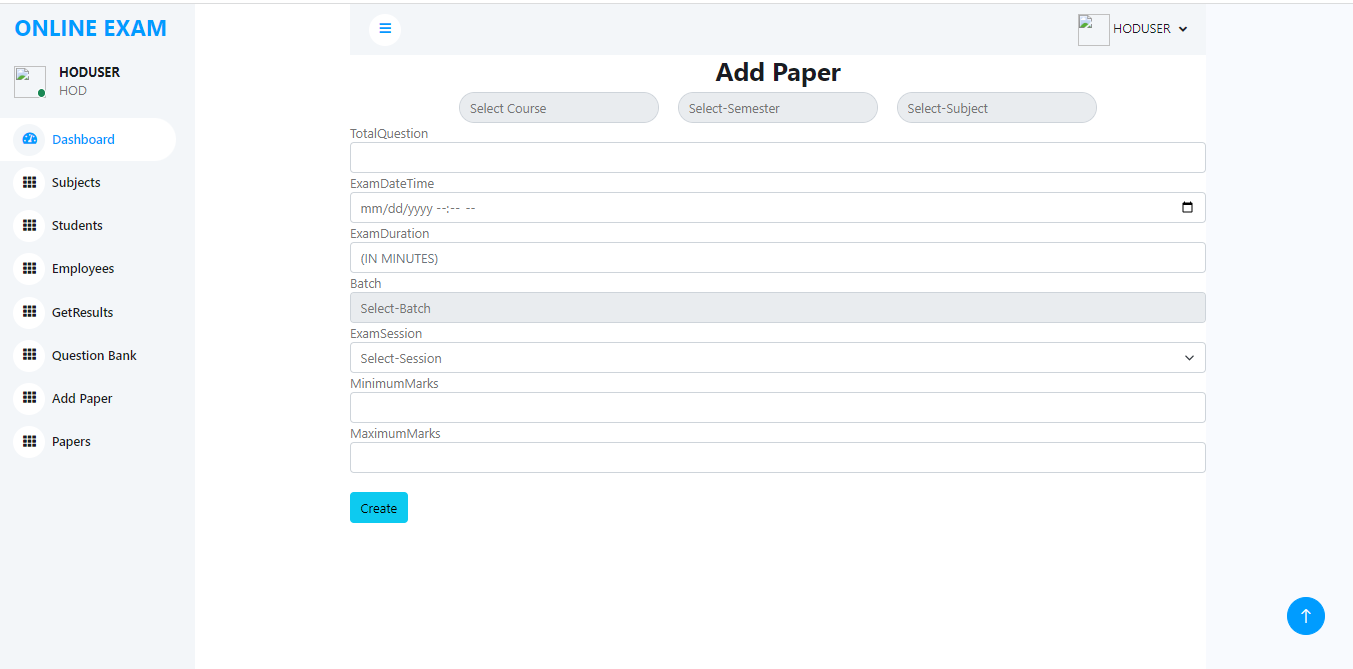
**13.4 Add Student**

****

**13.5 Question Bank**

****

**13.6 Add Paper**

****

**17.7 Exam**

****

Chapter 14

**BIBLIOGRAPHY**

**14. REFERENCES**

**14.1 BOOKS:**

1. Pressman, Roger S., Software Engineering, fifth edition, McGraw-Hill Higher Education, 2001.
2. Elmasri | Navathe, Fundamentals of Database Systems, third edition, Addison-Wesley, 2000.
3. Professional ASP.NET 4.0 in C# and VB by Bill Evgen, Scott Hanselman, Devin Rader.
4. Beginning ASP.NET 4.0 in C# by Matthew MacDonald.

**14.2**  **WEBSITES:**

1. [www.google.com](http://www.google.com/)
2. [www.wikipedia.com](http://www.wikipedia.com/)
3. [www.w3schools.com](http://www.w3schools.com/)
4. [www.aspforums.net](http://www.aspforums.net/)
5. [www.forums.tutorialized.com](http://www.forums.tutorialized.com/)

6. [www.youtube.com](http://www.youtube.com/)

1. [https://www.tutorialspoint.com/.Net/,Net\_introduction](https://www.tutorialspoint.com/.Net/%2CNet_introduction)
2. [https://www.geeksforgeeks.com/.Net/,Net\_introduction](https://www.geeksforgeeks.com/.Net/%2CNet_introduction)
3. System Analysis and Design by Elias M Awad
4. Software Engineering by Roger S Presman, TataMcgrawHill 2004
5. C# by C.Xavier Wrox Publication 2006
6. SQL Tips & Techniques by Kohrad King Contributing Editor Kris James, ph.D:MBA Microsoft Publications 2007
7. C# 2008 and the .NET 3.5 Platform, by Andrew W, Tyrolese, Works Publications 2007
8. Microsoft Visual C# 2008 Step by Step by John Sharp, Wrox Publications 2007
9. Illustrated C# 2008 by Daniel M. Solis Wrox Publications 2008