ARDENT®
"Drives You to Industry"

Industrial Internships 2024

is now approved by the

National Educational Alliance for Technology (NEAT)

cell of

All India Council of Technical Education (AICTE)

ISO 9001 2015 CERTIFIED

UDYAM REGISTRATION MSME

N·E·A·T
प्रौद्योगिकी के लिए राष्ट्रीय शैक्षणिक सहयोग
National Educational Alliance for Technology

Campus Recruitment
Training Program

ONLINE EXAM PANEL

Under guidance of

Mr . Ashraf Ali

A

Project Report

Submitted Partial Fulfilment Of The Requirement

For The Award Of the

Bachelor Of Technology

Project Carried Out At

ARDENT
COMPUTECH PVT. LTD.
High-End Technology Training and Project

Ardent ComputechPvt Ltd (An ISO 9001: 2008 Certified)

132,GroundFloor,SDFBuilding,

GPBlock,SectorV,Bidhannagar, Kolkata

,WestBengal,Kolkata -700091

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information of approval in any respect and will be summarily rejected.)

1. Name of the Student With Group:

1: Rajasree Laha
2: Md Afsar
3: Abdul Muksit
4:Raj Kumbhakar

2. Title of the Project: **Online Exam Panel**:

3. Name and Address of the Guide: **Mr . Ashraf Ali**

Sr. Subject Matter Expert
Technical Head(MERN)
Module #132,Ground Floor, SDF Building,
GP Block, Sector V, Bidhannagar, Kolkata,
West Bengal, Kolkata - 700091

4. Educational Qualification of the Guide:

Ph.d*	M.Tech	B.Tech*/B.E*	MCA*	M.Sc*

5. Working and Teaching experience of the Guide : .....Years
6. Software used in the Project: a. Visual Studio Code

1.
2.
3.
4.

Signature of the Guide
Date :
Mr. Ashraf Ali
Subject Matter Expert.
Signature, Designation, Stamp of the Project Proposal Evaluator

Signature of the Students
Date :

For the office use only

Approved	Not Approved

Project Responsibility Form

Sl	Name of Member	Responsibility
1	Rajasree Laha	Project Leader, Frontend,Coding, Face Login Implementation, Backend,DB Implementation
2	Md Afsar	Frontend, Coding, Debugging, Backend Review Project Documentation,
3	Abdul Muksit	Frontend, Coding, Debugging, Admin Panel, SystemAnalysis.
4	Raj Kumbhakar	Coding Rechecking Backend Debugging, Project Documentation, SystemAnalysis.

Self Certificate

This is to certify that the dissertation/project proposal entitled **“Online Exam Panel”** is done by us, is an Authentic work carried out for the partial fulfilment of the requirements for the award of the certificate of internship of MERN stack under the guidance of **Mr. Ashraf Ali**. The matter embodied in this project work has not been submitted earlier for award of any certificate to the best of our knowledge and belief.

Name of the Student :

- **1: Rajasree Laha**
- **2: Md Afsar**
- **3: Abdul Muksit**
- **4:Raj Kumbhakar**

Signature of the students :

1)

2)

3)

4)

Certificate by Guide

This is to certify that this project entitled **“Online Exam Panel”** submitted in partial fulfilment of the certificate of Bachelor of Computer Application through **Ardent Computech Pvt Ltd**, done by the Group Members

- **1: Rajasree Laha**
- **2: Md Afsar**
- **3: Abdul Muksit**
- **4:Raj Kumbhakar**

is an authentic work carried out under my guidance & best of our knowledge and belief..

1)

2)

3)

4)

Signature of the students
Date :

Signature of the Guide
Date :

Certificate of Approval

This is to certify that this proposal of the Minor project, entitled **“Online Exam Panel”** is a record of bona-fide work, carried out by : 1. Rajasree Laha, 2. Md Afsar, 3. Abdul Muksit, 4.Raj Kumbhakar, under my supervision and guidance through **Ardent Computech Pvt Ltd**. In my opinion, the report in its present form is in partial fulfilment of all the requirements. In fact, it has attained the standard necessary for submission. To the best of my knowledge, the results embodied in this report are original in nature and worthy of incorporation in the present version of the report for Internship For Bachelor of Technology.

Guide/Supervisor

Mr. Ashraf Ali

Subject Matter Expert & Technical Head (MERN)
Ardent Computech Pvt Ltd (An ISO 9001:2008 Certified) Module #132, Ground Floor, SDF Building, GP Block, Sector V, Bidhannagar, Kolkata, West Bengal, Kolkata - 700091

Head of the Department of
Computer Science and Engg..
(Aliah University)

1. ARDENT COMPUTECH PVT.LTD.

Ardent Computech Private Limited is an ISO 9001-2008 certified Software Development Company in India. It has been operating independently since 2003. It was recently merged with ARDENT TECHNOLOGIES.

Ardent Technologies

ARDENT TECHNOLOGIES is a Company successfully providing its services currently in the UK, USA, Canada and India. The core line of activity at ARDENT TECHNOLOGIES is to develop customised application software covering the entire responsibility of performing the initial system study, design, development, implementation and training. It also deals with consultancy services and Electronic Security systems. Its primary clientele includes educational institutes, entertainment industries, resorts, theme parks, service industry, telecom operators, media and other business houses working in various capacities.

Ardent Collaborations

ARDENT COLLABORATIONS, the Research Training and Development Department of ARDENT COMPUTECH PVT LTD is a professional training Company offering IT enabled services & industrial trainings for B-Tech, MCA, BCA, MSc and MBA freshers and experienced developers/programmers in various platforms. Summer Training / Winter Training / Industrial training will be provided for the students of B.TECH, M.TECH, MBA and MCA only. Deserving candidates may be awarded stipends, scholarships and other benefits, depending on their performance and recommendations of the mentors.

Associations

Ardent is an ISO 9001:2008 company. It is affiliated to National Council of Vocational Training (NCVT), Directorate General of Employment & Training (DGET), Ministry of Labor & Employment, and Government of India.

2 .INTRODUCTION

Welcome to our innovative **Online Exam Panel**, a cutting-edge solution designed to revolutionise the way users experience digital examinations. In a world characterised by the need for secure and efficient testing environments, our platform stands as a beacon of reliability, providing a seamless and efficient avenue for conducting exams online. With a user-friendly interface and a comprehensive set of features, our system allows educators to create and manage exams effortlessly, while ensuring students have a smooth and fair testing experience. Leveraging advanced technology, including face recognition for secure Admin login, our platform offers a robust and secure environment tailored to meet the needs of modern educational institutions.

At the heart of our system is a commitment to simplicity and accessibility. Users can effortlessly navigate through a visually appealing interface, explore and participate in exams with just a few clicks. Real-time exam updates ensure that students and educators have access to the latest information, empowering both exam takers and administrators. Our system is designed to make the online examination process intuitive and efficient, fostering a seamless experience for all users.

Security is paramount, and our platform employs robust authentication mechanisms to safeguard user information. The integration of advanced security measures, including face recognition for Admin login, enhances the overall examination experience, ensuring that only authorised users can access sensitive data. Our system is designed to prevent cheating and protect exam integrity, offering users a secure and trustworthy environment for all online examinations.

2A.OBJECTIVE

The primary objective of our Online Exam Panel is to provide a streamlined and convenient platform that transforms the traditional way of conducting exams and managing educational assessments. We aim to bridge the gap between educators and students, offering a modern solution that enhances the overall examination experience for users.

First and foremost, our system seeks to simplify the exam creation and administration process for educators. Through an intuitive and user-friendly interface, administrators and teachers can effortlessly create exams, manage question banks, and monitor exams in real-time. The goal is to save time and eliminate the complexities associated with traditional paper-based exams, empowering educators to focus on delivering quality education.

For students, our system aims to optimise the examination experience. We provide a centralised platform for exam management, allowing students to access exams securely, receive instant feedback, and track their performance seamlessly. This not only enhances student satisfaction but also improves their overall learning experience.

Ultimately, our Online Exam Panel strives to redefine the way people interact with the examination process, making the entire experience more efficient, secure, and accessible.

2.B.SCOPE

The scope of our Online Exam Panel is expansive, addressing the evolving demands of modern educational institutions and the dynamic nature of the learning environment. This system caters to a broad spectrum of users, including students seeking a seamless and secure exam experience, as well as educators aiming to enhance their exam management and assessment processes.

For students, the scope lies in the convenience of accessing exams online, receiving instant feedback, and tracking their performance in real-time. It offers personalised exam schedules, secure login features, and a smooth overall experience. Educators benefit from efficient exam creation, real-time monitoring, and automated grading processes. The system's scalability and adaptability ensure it remains relevant in an ever-changing educational landscape, making it a crucial tool for enhancing the efficiency and accessibility of online examinations in today's digital era.

SYSTEM ANALYSIS

3.A.IDENTIFICATION OF NEED

System analysis is a process of gathering and interpreting facts, diagnosing problems, and providing the information needed to recommend improvements to the system. It is a problem-solving activity that requires intensive communication between system users and developers. System analysis or study is a crucial phase in the development of any system, including our Online Exam Panel. During this phase, every detail of the system is meticulously analysed.

The system is viewed as a whole, with input and output processes identified. The outputs, such as exam results and performance reports, are traced back to the various processes involved in the system. System analysis involves understanding the problem, identifying key variables, analysing and synthesising various factors, and determining an optimal or satisfactory solution for the system.

Various techniques, such as interviews and questionnaires, are employed to conduct a detailed study of the processes involved. The data collected through these methods is thoroughly scrutinised to reach a conclusion about how the system functions. This phase defines the existing system and identifies any problem areas.

Once problem areas are identified, the system designer functions as a problem solver, developing proposals to address the system's challenges. These proposals are then compared with the existing system, and the best solution is selected. The proposal is presented to the user for approval, and revisions are made based on user feedback. This iterative process continues until the user is fully satisfied with the proposal, ensuring the system meets the needs of all stakeholders.

3.B.FEASIBILITY STUDY

A Feasibility Study is a comprehensive assessment conducted to evaluate the practicality and viability of a proposed project, such as the development of an Online Exam Panel. This study aims to determine whether the project is technically, financially, operationally, and economically feasible. It involves an in-depth analysis of various factors, including system requirements, technological capabilities, legal considerations, and available resources.

During a Feasibility Study, potential risks and challenges associated with the Online Exam Panel are identified, and potential solutions are explored. The study helps stakeholders make informed decisions by providing insights into the project's potential success or failure. Key components of a Feasibility Study include a technical requirements assessment, financial projections, operational considerations, and risk assessment.

Ultimately, a well-conducted Feasibility Study serves as a crucial foundation for decision-making, guiding stakeholders in deciding whether to proceed with the development of the Online Exam Panel or explore alternative options. This process ensures that all challenges and opportunities are thoroughly understood before moving forward.

3.C.WORKFLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirements of the system. It is meant for use by the developers and will be the basic during the testing phase. Any changes made to the requirements in the future will have to go through a formal change approval process. The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In the waterfall model phases do not overlap

❖ Waterfall Model design :

The waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure the success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In the Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

❖ Iterative Waterfall Design :

❑ Definition:

Ultimately, a well-conducted Feasibility Study serves as a crucial foundation for decision-making, guiding stakeholders in deciding whether to proceed with the development of the Online Exam Panel or explore alternative options. This process ensures that all challenges and opportunities are thoroughly understood before moving forward.

The sequential phases in Iterative Waterfall model are:

➢ **Requirement Gathering and analysis :** and analysis: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

➢ **System Design :** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

➢ **Implementation :** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

➢ **Integration and Testing :**All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

➢ **Deployment of system:**Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.

➢ **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment. All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Iterative Waterfall Model". In this model phases do not overlap.

❑ **Advantages**:

1 .**Flexibility**: Iterations permit adjustments based on feedback.

2 .**Early Delivery :**: Partial systems can be delivered incrementally.

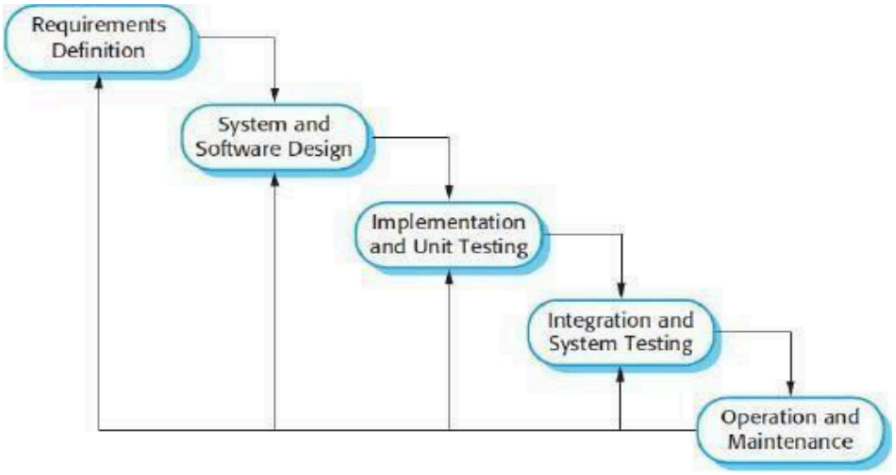
3 .**Risk Management** :Identifying and addressing issues early in the process .

❑ **<u>Disadvantages</u>**:

1 .**Increased Complexity**: Iterations permit adjustments based on feedback.

2 ..**Potential For Scope Creep :** Partial systems can be delivered incrementally.

3 .**Resource Intensive :** Identifying and addressing issues early in the process .



❑ **<u>Applications</u>** :

The Iterative Waterfall Model is suitable for projects with evolving or unclear requirements. It is commonly used in software development projects where regular feedback and refinement are essential. Additionally, it is applicable in scenarios where partial system delivery is beneficial, allowing stakeholders to assess progress and make adjustments.

3.D.STUDY OF THE SYSTEM

Modules :

The modules used in this software are as follows:

Login:

1. **User Login:** Here, users (students or educators) will log in to access exams and manage their accounts.
2. **Admin Login:** Here, the admin will log in using the face recognition technique to oversee exam activities, manage users, and handle the entire database.

Signup:

1. **Register:** Here, users (students or educators) will register to access the exam panel.

Home :

1. **Home:** This page is the main interface, allowing users to navigate to various sections of the exam panel.

About Us: This page will provide details about the platform and its developers.

Admin Interface :

1. **User Database:** Here, the admin can view and manage the details of all registered users.
2. **Exam Database:** Here, the admin can view and manage the details of all exams.

User Interface :

1. **Search for Exams:** Users (students) can search for and access their scheduled exams.
2. **Exam Results:** Here, users can view their completed exams and results.
3. **Account:** Here, users can manage their personal details and account settings after logging in.

3.E.INPUT AND OUTPUT

The main inputs , outputs and the major function the details are :

➢ **INPUT :**

1 . **User Login**:User can login by giving his credentials as input on the login page. (Same for Admin but Admin Can also Login using Face Login technique) .

2 . **Search Functionality**: Users can enter their preferred exam subjects and test names in the search bar.

3 . **Update Details:** : Users can enter their preferred exam subjects and test names in the search bar.

➢ **OUTPUT:**

1 . **User View:** Users can view the interface showing their exam details, personal information, and any previous test results.

2 . **Admin View:**Admins can access and view all databases, including user profiles, exam details, test results, and system logs.

3 . **Exam Management:** Exam managers can view the list of available exams, the list of scheduled exams, and update exam details or schedules based on availability.

3.F.SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and nonfunctional requirements. The software requirements specification document enlists enough necessary requirements that are required for the project development. To derive the requirements we need to have a clear and thorough understanding of the project to be developed. This is prepared after detailed communication with the project team and the customer.

The developer is responsible for:-

1. **Developing the System:** Creating the Online Exam Panel that meets the Software Requirements Specification (SRS) and addresses all the system's needs.
2. **System Demonstration and Installation:** Demonstrating the Online Exam Panel and installing it at the client's location after successful acceptance testing.
3. **Documentation Submission:** Providing the necessary user manual that describes the system interfaces and includes comprehensive documentation for the Online Exam Panel.
4. **User Training:** Conducting any required training sessions for users and administrators to effectively operate the Online Exam Panel.
5. **System Maintenance:** Providing system maintenance and support for a period of one year after installation.

Functional Requirements :

User Registration and Authentication :

1. Users should be able to create accounts securely.

2. The system should authenticate users and manage login sessions, including secure admin logins using face recognition.

Browse and Search:

1. Users should be able to browse and search for exams based on subjects, exam types, or other filters.

2. The system should authenticate users and manage login sessions, including secure admin logins using face recognition.

Exam Display:

1. Each exam should have detailed and up-to-date information, including exam schedules, duration, and instructions.

2. Users should be able to view exam descriptions and any related materials, such as sample questions or study guides.

Exam Registration:

1. Users should be able to register for exams.

2. Users should be able to specify any required details, such as preferred time slots, exam centre locations, or special accommodations.

Non-Functional Requirements :

Performance:

1. The Online Exam Panel should respond to user actions promptly, with minimal latency during exam operations, such as loading questions or submitting answers.

2. The platform must efficiently handle increased user loads, especially during peak exam periods, without significant performance degradation.

Reliability:

1. The Online Exam Panel should be consistently available and accessible to users, minimising downtime, especially during scheduled exams.

2. The system should gracefully handle errors or failures, ensuring uninterrupted service during exams, with mechanisms to recover from unexpected issues.

Security:

1. Ensure the secure transmission of sensitive information, such as user credentials, exam data, and personal details.

2. Implement robust mechanisms to authenticate users, including face login for admins, and authorise access to specific features based on user roles.

Usability:

1. The Online Exam Panel should have a user-friendly interface that is easy to navigate, allowing users to focus on their exams without confusion.

2. Ensure the platform is accessible to users with disabilities, complying with relevant standards, such as providing screen reader support and keyboard navigation.

❑ **Hardware Requirements:**

1 . Computer hasIntel I5 Processor

2 . 8 GB RAM

3 . DVD-ROM Drive

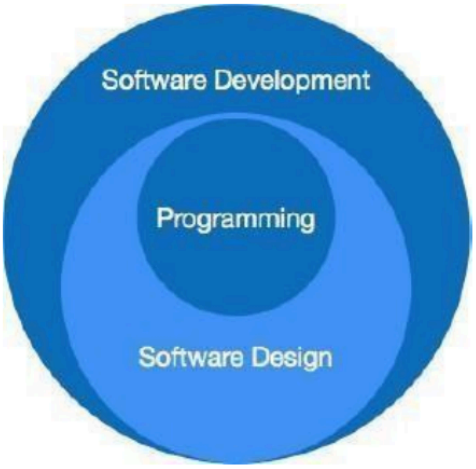
❑ **Software Requirements:**

4 . Windows 11 OS

- ☐ **. Visual Studio Code**

SOFTWARE ENGINEERING PARADIGM APPLIED

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another.



The programming paradigm is a subset of the Software design paradigm which is further a subset of the Software development paradigm. There are two levels of reliability. The first is meeting the right requirements. A careful and thorough systems study is needed to satisfy this aspect of reliability. The second level of systems reliability involves the actual work delivered to the user. At this level, the system's reliability is interwoven with software engineering and development.

There are three approaches to reliability.

1. . **Error avoidance** : Prevents errors from occurring in software.

2. **Error detection and correction:** : In this approach, errors are recognized whenever they are encountered, and correcting the error by the effect of the error of the system does not fail.

3. . **Error tolerance** : In this approach, errors are recognized whenever they occur, but enables the system to keep running through degraded performance or Applying values that instruct the system to continue the process.

SYSTEM DESIGN

4.A.DATA FLOW DIAGRAM

A **data flow diagram (DFD)** is a graphical representation of the "**flow**" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

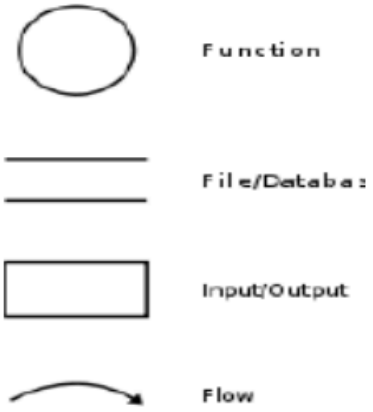
DFDs can also be used for the visualisation of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modelled. The Level 1 DFD shows how the system is divided into subsystems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present for the system to do its job and shows the flow of data between the various parts of the system. Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users can visualise how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's data flow diagrams can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately affects the structure of the whole system from order to dispatch to report.

How any system is developed can be determined through a data flow diagram model. In the course of developing a set of levelled data flow diagrams, the analyst/designer is forced to address how the system may be decomposed into component sub-systems and to identify the transaction data in the data model. Data flow diagrams can be used in both the Analysis and Design phase of the SDLC. There are different notations to draw data flow

diagrams. defining different visual representations for processes, data stores, data flow, and external entities.

❑ DFD Notation :



❑ DFD EXAMPLE :

Steps to Construct Data Flow Diagram :-

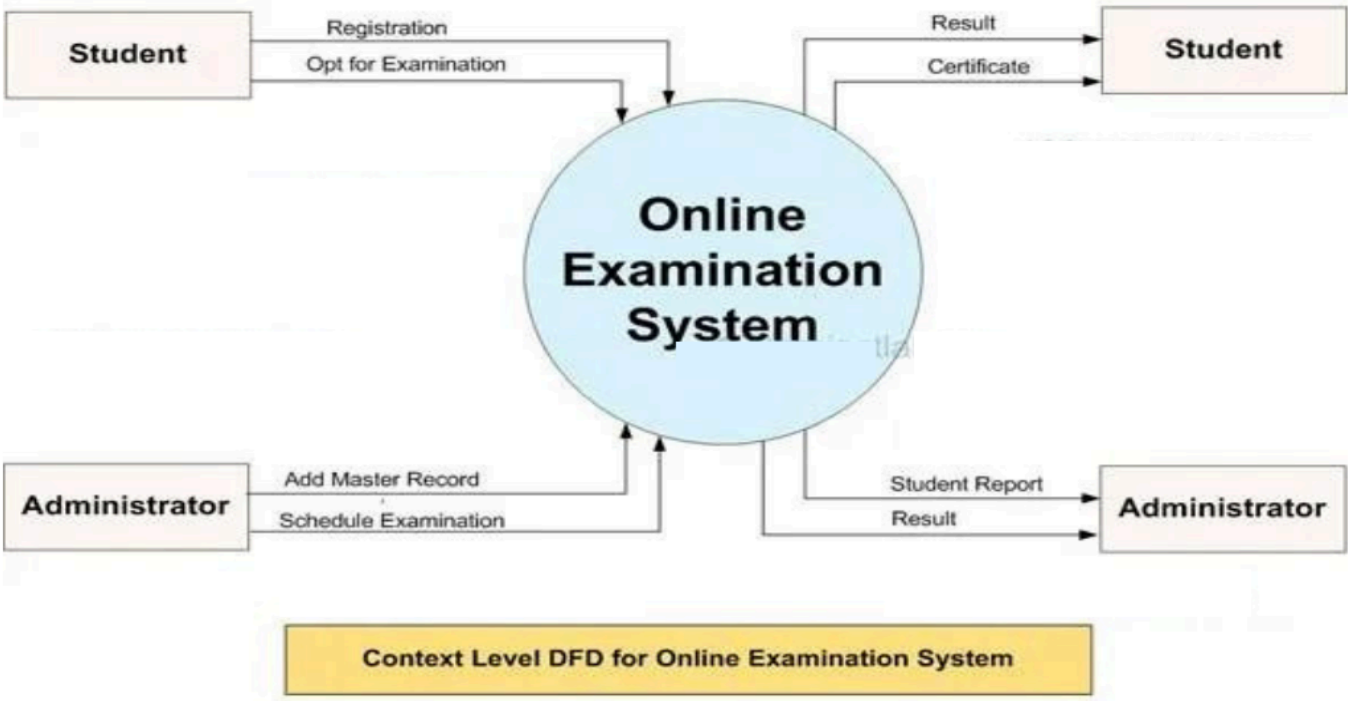
Four Steps are generally used to construct a DFD.

- Process should be named and referred for easy reference. Each name should be representative of the reference.
- The destination of flow is from top to bottom and from left to right.
- When a process is distributed into lower-level details they are numbered.
- The names of data stores, sources, and destinations are written in capital letters.

Rules for constructing a Data Flow Diagram:-

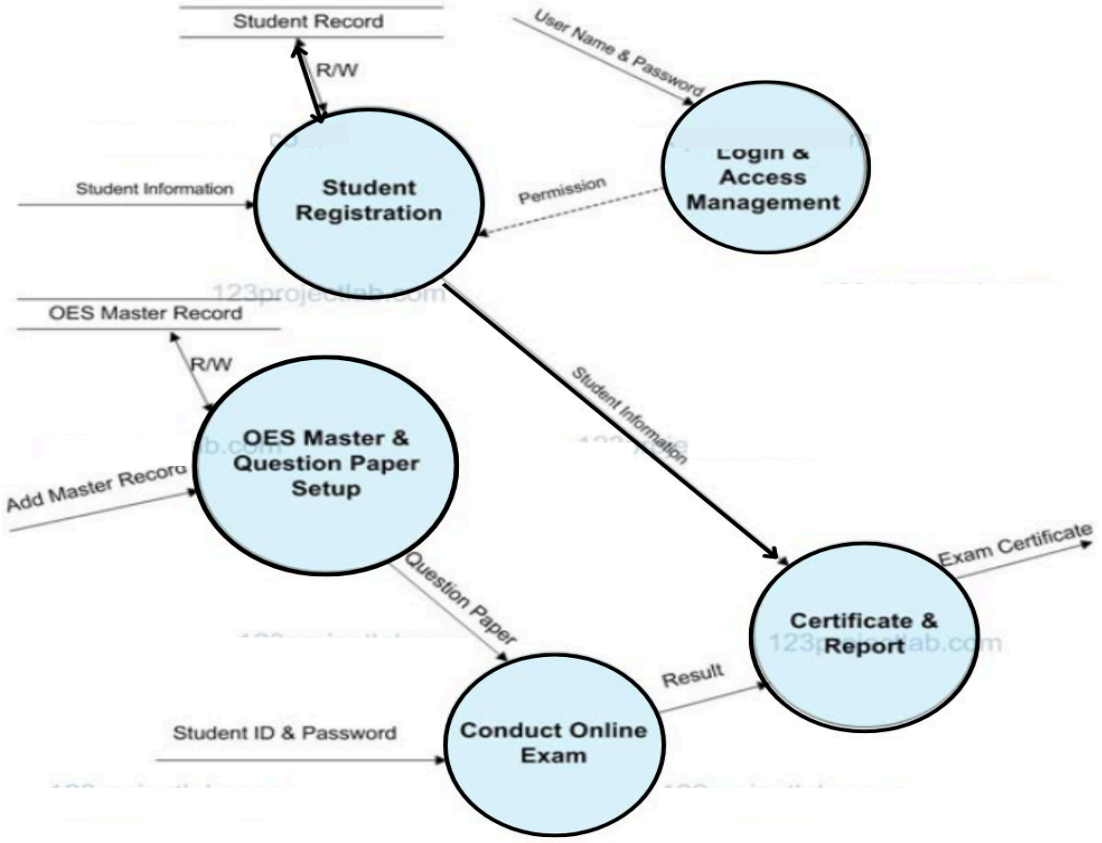
- Arrows should not cross each other.
- Squares, Circles, and Files must bear a name.
- Decomposed data flow squares and circles can have the same names.
- Draw all data flow around the outside of the diagram.

LEVEL 0 DFD OR CONTEXT DIAGRAM :



Context Level DFD for Online Examination System

LEVEL 1 DFD :



Level 1 DFD for Online Exam Panel

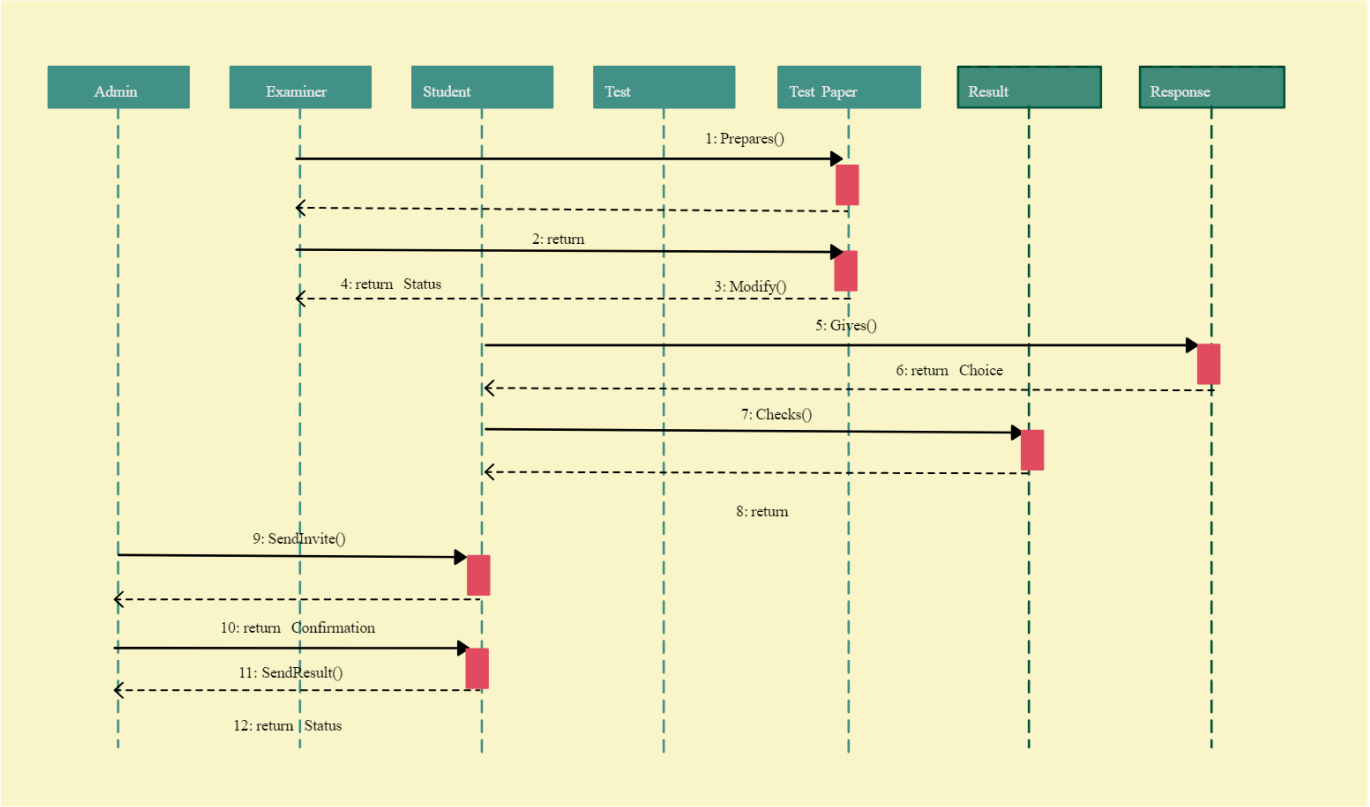
4.B.SEQUENCE DIAGRAM

A **Sequence diagram** is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realisations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A sequence diagram is the most common kind of **interaction diagram**, which focuses on the **message** interchange between several lifelines. A sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines. The following nodes and edges are typically drawn in a **UML sequence**

diagram: **lifeline**, **execution specification**, **message**, fragment, interaction,**state invariant**, **continuation**, and **destruction occurrence**.



Sequence Diagram Of Online Exam Panel

4.C.USE CASE DIAGRAM

A Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. So only static behaviour is not sufficient to model a system, rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and a use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction. These internal and external agents are known as actors. So use case diagrams consist of actors, use cases, and their relationships. The diagram is used to model the system/subsystem of an application. A single-use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used. The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Because the other four diagrams (activity, sequence, collaboration, and State chart) are also having the same purpose. So we will look into some specific purpose that will distinguish it from the other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analysed to gather its functionalities use cases are prepared and actors are identified. Now when the initial task is complete use case diagrams are modelled to present the outside view. So in brief, the purposes of use case diagrams can be as follows:

➢ Used to gather requirements of a system.
➢ Used to get an outside view of a system.
➢ Identify external and internal factors influencing the system.
➢ Show the interaction among the requirements actors.


OnlineExaminationSystem
System
Register
(include)
Fill Online Form
Answer Questions
(extend)
Submit Paper
(extend)
View Exam Results
student
Login
(include)
Authentication
Add Qs
(extend)
Update Qs
(extend)
Set Questions Paper
(extend)
View Qs
(extend)
Delete Qs
Review Students Application
Declare Results
(extend)
Email Results
faculty members

Schema Design

The schema is an abstract structure or outline representing the logical view of the database as a whole. Defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

1. Users Table

Stores information about all users, including students and admins.

Column Name	Data Type	Description
user_id	INT (Primary Key)	Unique identifier for each user
username	VARCHAR(255)	Username for login
password_hash	VARCHAR(255)	Hashed password
email	VARCHAR(255)	User's email address
role	ENUM('student', 'admin')	Role of the user
face_data	BLOB	Biometric data for face login (for admins)
created_at	TIMESTAMP	Account creation timestamp

2. Exams Table

Stores information about the exams.

Column Name	Data Type	Description
exam_id	INT (Primary Key)	Unique identifier for each exam
exam_name	VARCHAR(255)	Name of the exam
subject	VARCHAR(255)	Subject of the exam
description	TEXT	Description of the exam
start_time	DATETIME	Scheduled start time of the exam
duration	INT	Duration of the exam in minutes
total_marks	INT	Total marks for the exam
created_by	INT (Foreign Key)	Admin who created the exam

3. Questions Table

Stores the questions for each exam.

Column Name	Data Type	Description
question_id	INT (Primary Key)	Unique identifier for each question
exam_id	INT (Foreign Key)	Associated exam's ID
question_text	TEXT	The question content
option_a	VARCHAR(255)	Option A
option_b	VARCHAR(255)	Option B
option_c	VARCHAR(255)	Option C
option_d	VARCHAR(255)	Option D
correct_option	ENUM('A', 'B', 'C', 'D')	Correct answer option
marks	INT	Marks allocated for the question

4. User_Exam Table

Associates users with the exams they are taking.

Column Name	Data Type	Description
user_exam_id	INT (Primary Key)	Unique identifier for each entry
user_id	INT (Foreign Key)	User ID
exam_id	INT (Foreign Key)	Exam ID
status	ENUM('registered', 'in-progress', 'completed')	Status of the exam for the user
score	INT	Score obtained by the user
started_at	DATETIME	Time when the user started the exam
completed_at	DATETIME	Time when the user completed the exam

5. Responses Table

Stores the user's answers to each question.

Column Name	Data Type	Description
response_id	INT (Primary Key)	Unique identifier for each response
user_exam_id	INT (Foreign Key)	User Exam ID
question_id	INT (Foreign Key)	Question ID
selected_option	ENUM('A', 'B', 'C', 'D')	The option selected by the user
is_correct	BOOLEAN	Whether the selected option is correct

6. Results Table

Stores final results for each exam.

Column Name	Data Type	Description
result_id	INT (Primary Key)	Unique identifier for each result
user_exam_id	INT (Foreign Key)	User Exam ID
total_score	INT	Total score obtained in the exam
result_status	ENUM('passed', 'failed')	Final result status
generated_at	DATETIME	Time when the result was generated

7. Logs Table

Stores logs for admin actions and significant system events.

Column Name	Data Type	Description
log_id	INT (Primary Key)	Unique identifier for each log
admin_id	INT (Foreign Key)	Admin ID
action	TEXT	Description of the action
log_time	DATETIME	Timestamp of the action

Database tables are essential because they organise and store data in a structured way, enabling efficient data retrieval, maintaining relationships between data, ensuring data integrity, and supporting scalability and security in applications like an Online Exam Panel.

Create Test:

```
import React, { useState, useEffect } from 'react'
import { Card, Col, Container, Form, FormLabel, Row } from 'react-bootstrap'


export default function CreateTest(props) {


    const [testForm, setTestForm] = useState({
        admin: {
            adminId: props.adminId
        },
        testName: "",
        questionList: []
    });


    const [questions, setQuestions] = useState([]);


    const submit = (e) => {
        console.log(testForm);
        e.preventDefault();
        props.saveTest(testForm);
    }


    useEffect(() => {
        setTestForm(prevTestForm => ({ ...prevTestForm, questionList:
questions }));
    }, [questions]);
```

```
    const handleFormChange = (index, event) => {
        event.preventDefault();
        let data = [...questions];
        data[index][event.target.name] = event.target.value;
        setQuestions(data);
    }

    const handleFormChoiceChange = (index, event, choiceNo) => {
        event.preventDefault();
        let data = [...questions];
        data[index]["choicesList"][choiceNo] = { choiceDesc:
event.target.value, ans: 0 };
        setQuestions(data);
    }

    const handleAnswerChange = (index, event) => {
        event.preventDefault();
        let data = [...questions];
        for (let i = 0; i < 4; i++) {
            data[index]["choicesList"][i] = {
...(data[index]["choicesList"][i]), ans: 0 };
        }
        data[index]["choicesList"][event.target.value - 1] = {
...(data[index]["choicesList"][event.target.value - 1]), ans: 1 };
        setQuestions(data);
    }
```

```
const addQuestions = (e) => {
    e.preventDefault();
    let newQues = {
        quesDesc: "",
        choicesList: [
            {
                choiceDesc: "",
                ans: 0
            },
            {
                choiceDesc: "",
                ans: 0
            },
            {
                choiceDesc: "",
                ans: 0
            },
            {
                choiceDesc: "",
                ans: 0
            }
        ]
    }

    setQuestions([...questions, newQues]);
}
```

```
    const removeFields = (e, index) => {
        e.preventDefault();
        let data = [...questions];
        data.splice(index, 1)
        setQuestions(data)
    }

    const openDashboard = (e) => {
        props.openAdminDashboard();
        e.preventDefault();
    }

    return (
        <Container className='my-5' style={{backgroundColor: 'rgba(255, 255, 255, 0.9)', overflowY: "scroll", maxHeight:"70vh"}}>
            <Form onSubmit={submit}>
                <Card className='my-3'>
                    <Card.Body>
                        <FormLabel htmlFor='testName'><h4>Test Name:</h4></FormLabel>
                        <Form.Control
                            type="text"
                            id="testName"
                            value={testForm.testName}
                            onChange={(e) => setTestForm({ ...testForm, testName: e.target.value })}
                            placeholder="Give a name to your test.."
                        />
```

```
                    </Card.Body>
                </Card>

                {questions.map((input, index) => {
                    return (<div key={index}>
                        <Card className='m-1 p-2'>
                            <Card.Header>
                                <Row className="justify-content-center">
                                    <Col md="auto">
                                        <p>{index + 1}.</p>
                                    </Col>
                                    <Col>
                                        <FormLabel
htmlFor='quesDesc'></FormLabel>
                                        <Form.Control
                                            type="text"
                                            id="quesDesc"
                                            name="quesDesc"
                                            value={input.quesDesc}
                                            onChange={event =>
handleFormChange(index, event)}
                                            placeholder="Enter question
text"
                                        /></Col>
                                    <Col md="auto">
                                        <button className="float-right mb-2
btn-sm btn-danger" onClick={(e) => removeFields(e,
index)}>Delete</button></Col>
                                </Row>
```

```
                            </Card.Header>
                            <Card.Body>
                                <Row>
                                    <Col>
                                        <Form.Control
                                            type="text"
                                            id="choice1"
                                            name="choice1"
value={input.choicesList[0].choiceDesc}
                                            onChange={event =>
handleFormChoiceChange(index, event, 0)}
                                            placeholder="Option 1 text"
                                        />
                                    </Col>
                                    <Col>
                                        <Form.Control
                                            type="text"
                                            id="choice2"
                                            name="choice2"
value={input.choicesList[1].choiceDesc}
                                            onChange={event =>
handleFormChoiceChange(index, event, 1)}
                                            placeholder="Option 2 text"
                                        />
                                    </Col>
                                </Row>
```

```
                                <Row>
                                    <Col>
                                        <Form.Control
                                            type="text"
                                            id="choice3"
                                            name="choice3"

value={input.choicesList[2].choiceDesc}
                                            onChange={event =>
handleFormChoiceChange(index, event, 2)}
                                            placeholder="Option 3 text"
                                        />
                                    </Col>
                                    <Col>
                                        <Form.Control
                                            type="text"
                                            id="choice4"
                                            name="choice4"

value={input.choicesList[3].choiceDesc}
                                            onChange={event =>
handleFormChoiceChange(index, event, 3)}
                                            placeholder="Option 4 text"
                                        />
                                    </Col>
                                </Row>


                            </Card.Body>
                            <Card.Footer>
```

```
                                <Row>
                                    <Col>
                                        <Form.Select className='bg-light'
onChange={event => handleAnswerChange(index, event)}>
                                            <option>Select Answer</option>
                                            <option value="1">1</option>
                                            <option value="2">2</option>
                                            <option value="3">3</option>
                                            <option value="4">4</option>
                                        </Form.Select>
                                    </Col>
                                </Row>
                            </Card.Footer>
                        </Card>
                    </div>
                    );
                })}
                <Row className='py-2 text-center'>
                    <Col>
                        <button className="btn btn-primary"
onClick={addQuestions}>Add More Question</button>
                    </Col>
                </Row>
                <Row className='py-2 text-center'>
                    <Col>
                        <button className="btn btn-primary" type='submit'
onClick={submit}>Save Test</button>
                    </Col>
```

```
                    <Col>

                        <button className="btn btn-danger"
onClick={openDashboard}>Cancel</button>

                    </Col>

                </Row>

            </Form>

        </Container> )}
```

Available Test :

```
import React from 'react';

import TestListItem from '../Test/TestListItem';

import { Container } from 'react-bootstrap';



export default function AvailableTest(props) {

  console.log(props.studentTestList);

  const testList = Object.keys(props.studentTestList).map((key) =>
props.studentTestList[key]);

  console.log(testList)



  const backToDashboard = (e) => {

    e.preventDefault();

    props.openDashboard();

}



  return (

    <Container className='my-5 bg-light pt-2' style={{maxWidth: "100vh",
overflowY: "scroll", maxHeight:"60vh"}}>

    <button type="button" className="close" onClick={backToDashboard}
aria-label="Close">

                        <span aria-hidden="true">&times;</span>

                    </button>

                    <br/>

                    <h4 className='text-center'>Available Tests</h4>

    <Container className="bg-light p-2">
```

```
        {testList.map((test) => {
          console.log(test.testId);
          return (
            <TestListItem key={test.testId} test={test} />
          )
        })}
    </Container>

    </Container>
  )
}
```

Students Profile :

```
import React, { useState } from 'react';
import { Card, Button, Container, Row, Col } from 'react-bootstrap';
import Form from 'react-bootstrap/Form';


export default function StudentProfile(props) {



    let formData = { ...props.student };
    const [student, setStudent] = useState({ ...props.student });
    const [isFormDisabled, setIsFormDisabled] = useState(true)


    const editForm = () => {
        setIsFormDisabled(false);
    }
```

```
    const resetForm = () => {
        setStudent({ ...formData });
        setIsFormDisabled(true);
    }

    const saveDetails = (e) => {
        e.preventDefault();
        props.saveStudentDetails(student);
        setIsFormDisabled(true);
    }

    const backToDashboard = (e) => {
        e.preventDefault();
        props.openDashboard();
    }

    return (
        <Container className='sm my-5'>
            <Card className='bg-light'>
                <Card.Header className="lg">
                    <b>User details</b>
                    <button type="button" className="close"
onClick={backToDashboard} aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </Card.Header>
```

```
                <Card.Body className='p-3'>
                    <Row className='mx-5 mb-3'>
                        <Col>
                            <Form.Label htmlFor="inputName">Full
Name</Form.Label>
                            <Form.Control
                                type="text"
                                id="inputName"
                                value={student.studentName}
                                onChange={(e) => setStudent({ ...student,
studentName: e.target.value })}
                                aria-describedby="Student Name"
                                disabled={isFormDisabled ? true : false}
                            />
                        </Col>
                        <Col>
                            <Form.Label htmlFor="inputEmail">Email
Address</Form.Label>
                            <Form.Control
                                type="text"
                                id="inputEmail"
                                value={student.email}
                                onChange={(e) => setStudent({ ...student,
email: e.target.value })}
                                aria-describedby="Email Address"
                                disabled={true}
                            />
                        </Col>
                    </Row>
```

```
                    <Row className='mx-5 mb-3'>
                        <Col>
                            <Form.Label htmlFor="inputMobno">Mobile
No.</Form.Label>
                            <Form.Control
                                type="text"
                                id="inputMobno"
                                value={student.mobNo}
                                onChange={(e) => setStudent({ ...student,
mobNo: e.target.value })}
                                aria-describedby="Mobile No"
                                disabled={isFormDisabled ? true : false}
                            />
                        </Col>
                        <Col>
                            <Form.Label
htmlFor="inputPass">Password</Form.Label>
                            <Form.Control
                                type="password"
                                id="inputPass"
                                value={student.pass}
                                onChange={(e) => setStudent({ ...student,
pass: e.target.value })}
                                aria-describedby="Mobile No"
                                disabled={isFormDisabled ? true : false}
                            />
                        </Col>
                    </Row>
                    <Row className='mx-5 mb-3'>
```

```
                        <Col>
                            <Form.Label
htmlFor="inputCity">City</Form.Label>
                            <Form.Control
                                type="text"
                                id="inputCity"
                                value={student.address.city}
                                onChange={(e) => setStudent({ ...student,
address: { ...student.address, city: e.target.value } })}
                                aria-describedby="Address City"
                                disabled={isFormDisabled ? true : false}
                            />
                        </Col>
                        <Col>
                            <Form.Label
htmlFor="inputState">State</Form.Label>
                            <Form.Control
                                type="text"
                                id="inputState"
                                value={student.address.state}
                                onChange={(e) => setStudent({ ...student,
address: { ...student.address, state: e.target.value } })}
                                aria-describedby="Address State"
                                disabled={isFormDisabled ? true : false}
                            />
                        </Col>
                    </Row>
                    <Row className='mx-5 mb-3'>
                        <Col>
```

```
                            <Form.Label
htmlFor="inputCountry">Country</Form.Label>

                            <Form.Control

                                type="text"

                                id="inputCountry"

                                value={student.address.country}

                                onChange={(e) => setStudent({ ...student,
address: { ...student.address, country: e.target.value } })}

                                aria-describedby="Address Country"

                                disabled={isFormDisabled ? true : false}

                            />

                        </Col>

                        <Col>

                            <Form.Label
htmlFor="inputZip">Zipcode</Form.Label>

                            <Form.Control

                                type="text"

                                id="inputZip"

                                value={student.address.zip}

                                onChange={(e) => setStudent({ ...student,
address: { ...student.address, zip: e.target.value } })}

                                aria-describedby="Address Zipcode"

                                disabled={isFormDisabled ? true : false}

                            />

                        </Col>

                    </Row>

                </Card.Body>

                <Row className='mx-5 mb-5 text-center'>

                    {isFormDisabled ? <Col>
```

```
                        <Button variant="primary" size="lg"
onClick={editForm}>Edit</Button>

                    </Col> : <><Col>

                        <Button variant="primary" size="lg"
onClick={saveDetails}>Save</Button>

                    </Col><Col>

                            <Button variant="primary" size="lg"
onClick={resetForm}>Cancel</Button>

                        </Col></>}

                </Row>



            </Card>

        </Container>

    )

}
```

Track Record :

```
import React from 'react'



export default function TrackRecords() {

  return (

    <div>TrackRecords</div>

  )

}
```

Test English :

```
import React, { useState, useEffect } from 'react';

import { Container, Row, Col, Button, Form, Badge, Card } from
'react-bootstrap';
```

```
import { useNavigate } from 'react-router-dom';


const questions = [

  {

    id: 'q1',

    question: 'Choose the correct form of the verb: "She ____ to the store yesterday."',

    options: ['goes', 'went', 'gone', 'going']

  },

  {

    id: 'q2',

    question: 'Select the correctly punctuated sentence:',

    options: ['Its raining cats and dogs.', 'It\'s raining cats and dogs.', 'It raining cats and dogs.', 'Its raining, cats and dogs.']

  },



  {

    id: 'q3',

    question: 'What is the synonym of "ubiquitous"?',

    options: ['rare', 'present everywhere', 'unique', 'distant']

  },

  {

    id: 'q4',

    question: 'Identify the grammatical error in the sentence: "He don\'t like to play basketball."',

    options: ['don\'t should be doesn\'t', 'play should be plays', 'basketball should be basketballs', 'no error']

  },

  {
```

```
    id: 'q5',
    question: 'Fill in the blank with the correct preposition: "She is keen
____ learning new languages."',
    options: ['in', 'on', 'at', 'to']
  }
];


const TestE = () => {
  const [currentQuestionIndex, setCurrentQuestionIndex] = useState(0);
  const [selectedOption, setSelectedOption] = useState(null);
  const [answers, setAnswers] =
useState(Array(questions.length).fill(null));
  const [isMarkedForReview, setIsMarkedForReview] =
useState(Array(questions.length).fill(false));
  const [timer, setTimer] = useState(10719); // Time in seconds (2 hours 58
minutes 39 seconds)
  const navigate = useNavigate();



 // Timer function
  useEffect(() => {
    const intervalId = setInterval(() => {
      setTimer(prevTime => {
        if (prevTime <= 0) {
          clearInterval(intervalId);
          return 0;
        }
        return prevTime - 1;
      });
```

```
    }, 1000);


    return () => clearInterval(intervalId);
  }, []);


  const handleOptionChange = (e) => {
    setSelectedOption(e.target.id);
  };


  const handleMarkForReview = () => {
    const updatedMarks = [...isMarkedForReview];
    updatedMarks[currentQuestionIndex] =
!isMarkedForReview[currentQuestionIndex];
    setIsMarkedForReview(updatedMarks);


    alert(isMarkedForReview[currentQuestionIndex] ? 'Question unmarked for
review.' : 'Question marked for review.');
  };
  const handleClearResponse = () => {


   setSelectedOption(null);
    alert('Response cleared.');
  };


  const handleSaveAndNext = () => {
    const updatedAnswers = [...answers];
    if (selectedOption !== null) {
      updatedAnswers[currentQuestionIndex] = selectedOption;
```

```
      alert('Response saved.');
    } else {
      // If no response is selected, mark as not answered
      if (updatedAnswers[currentQuestionIndex] === null) {
        alert('Question not answered.');
      }
    }
    setAnswers(updatedAnswers);

    // Move to the next question
    if (currentQuestionIndex < questions.length - 1) {
      setCurrentQuestionIndex(currentQuestionIndex + 1);
      setSelectedOption(updatedAnswers[currentQuestionIndex + 1]);
    } else {
      alert('You have completed the test.');
      navigate('/test/summary'); // Navigate to a summary page or wherever
you want
    }
  };



 const formatTime = (seconds) => {
    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);
    const secs = seconds % 60;
    return `${hours} Hours ${minutes} Minutes ${secs} Seconds`;
  };
```

```
  const currentQuestion = questions[currentQuestionIndex];


  const getButtonVariant = (index) => {
    if (answers[index] !== null) {
      return isMarkedForReview[index] ? 'warning' : 'success'; // Answered and marked for review vs. answered
    } else {
      return isMarkedForReview[index] ? 'info' : 'light'; // Not answered but marked vs. not answered
    }
  };


  return (
    <Container className="my-4">
      <Card>
        <Card.Header>
          <h5>English Test</h5>
        </Card.Header>
        <Card.Body>
          {/* Question Section */}
          <Row className="mb-4">


       <Col>
              <h5>Question No. {currentQuestionIndex + 1}</h5>
              <p>{currentQuestion.question}</p>
              <Form>
                {currentQuestion.options.map((option, index) => (
```

```
                <Form.Check
                  key={index}
                  type="radio"
                  label={option}
                  name={`question${currentQuestionIndex}`}
                  id={`option${index}`}
                  checked={selectedOption === `option${index}`}
                  onChange={handleOptionChange}
                />
              ))}
            </Form>
          </Col>
        </Row>

        {/* Action Buttons */}
        <Row className="mb-4">
          <Col>
            <Button
              variant={isMarkedForReview[currentQuestionIndex] ? "warning" : "primary"}
              className="me-2"
              onClick={handleMarkForReview}
            >

  {isMarkedForReview[currentQuestionIndex] ? "Unmark for Review" : "Mark for Review"}
            </Button>
```

```
              <Button variant="danger" className="me-2"
onClick={handleClearResponse}>Clear Response</Button>

              <Button variant="success" onClick={handleSaveAndNext}>Save &
Next</Button>

            </Col>

          </Row>



          {/* Timer and Question Palette */}

          <Row>

            <Col md={6} className="mb-4">

              <div className="border p-3">

                <h6>Time Remaining</h6>

                <div className="text-center">

                  <h5>{formatTime(timer)}</h5>

                </div>

              </div>

            </Col>

            <Col md={6} className="mb-4">

              <div className="border p-3">

                <h6>Question Palette</h6>

                <div className="d-flex flex-wrap">

                  {questions.map((_, index) => (

                    <Button

                      key={index}

                      variant={getButtonVariant(index)}

                      className="m-1"



        onClick={() => {
```

```
                        setCurrentQuestionIndex(index);
                        setSelectedOption(answers[index]);
                      }}
                    >
                      {index + 1}
                    </Button>
                  ))}
                </div>
                <div className="mt-3">
                  <Badge pill bg="success" className="me-2">Answered</Badge>
                  <Badge pill bg="danger" className="me-2">Not
Answered</Badge>
                  <Badge pill bg="warning" className="me-2">Marked</Badge>
                  <Badge pill bg="info" className="me-2">Answered & Marked
for Review</Badge>
                </div>
              </div>
            </Col>
          </Row>
        </Card.Body>
      </Card>
    </Container>
  );
};

export default TestE;
```

Test Math :

```
import React, { useState, useEffect } from 'react';

import { Container, Row, Col, Button, Form, Badge, Card } from
'react-bootstrap';

import { useNavigate } from 'react-router-dom';



const questions = [

  {

    id: 'q1',

    question: 'What is the result of 5 + 3?',

    options: ['6', '7', '8', '9']

  },

  {

    id: 'q2',

    question: 'Solve for x: 2x - 4 = 10',

    options: ['x = 4', 'x = 5', 'x = 6', 'x = 7']

  },

  {

    id: 'q3',

    question: 'What is the area of a rectangle with length 8 cm and width 5
cm?',

    options: ['30 cm²', '35 cm²', '40 cm²', '45 cm²']

  },

  {

    id: 'q4',

    question: 'Find the square root of 144.',

    options: ['10', '12', '14', '16']

  },
```

```
  {
    id: 'q5',

  question: 'What is the value of 7 × (6 - 2)?',
    options: ['20', '24', '28', '32']
  }
];

const TestM = () => {
  const [currentQuestionIndex, setCurrentQuestionIndex] = useState(0);
  const [selectedOption, setSelectedOption] = useState(null);
  const [answers, setAnswers] =
useState(Array(questions.length).fill(null));
  const [isMarkedForReview, setIsMarkedForReview] =
useState(Array(questions.length).fill(false));
  const [timer, setTimer] = useState(10719); // Time in seconds (2 hours 58
minutes 39 seconds)
  const navigate = useNavigate();

  // Timer function
  useEffect(() => {
    const intervalId = setInterval(() => {
      setTimer(prevTime => {
        if (prevTime <= 0) {
          clearInterval(intervalId);
          return 0;
        }
        return prevTime - 1;
```

```
      });

    }, 1000);


    return () => clearInterval(intervalId);


      }, []);


  const handleOptionChange = (e) => {

    setSelectedOption(e.target.id);

  };


  const handleMarkForReview = () => {

    const updatedMarks = [...isMarkedForReview];

    updatedMarks[currentQuestionIndex] =
!isMarkedForReview[currentQuestionIndex];

    setIsMarkedForReview(updatedMarks);


    alert(isMarkedForReview[currentQuestionIndex] ? 'Question unmarked for
review.' : 'Question marked for review.');

  };


  const handleClearResponse = () => {

    setSelectedOption(null);

    alert('Response cleared.');

  };


  const handleSaveAndNext = () => {

    const updatedAnswers = [...answers];
```

```
    if (selectedOption !== null) {
      updatedAnswers[currentQuestionIndex] = selectedOption;
      alert('Response saved.');
    } else {
      // If no response is selected, mark as not answered
      if (updatedAnswers[currentQuestionIndex] === null) {

       alert('Question not answered.');
      }
    }
    setAnswers(updatedAnswers);

    // Move to the next question
    if (currentQuestionIndex < questions.length - 1) {
      setCurrentQuestionIndex(currentQuestionIndex + 1);
      setSelectedOption(updatedAnswers[currentQuestionIndex + 1]);
    } else {
      alert('You have completed the test.');
      navigate('/test/summary'); // Navigate to a summary page or wherever
you want
    }
  };

  const formatTime = (seconds) => {
    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);
    const secs = seconds % 60;
```

```
    return `${hours} Hours ${minutes} Minutes ${secs} Seconds`;
  };

  const currentQuestion = questions[currentQuestionIndex];

  const getButtonVariant = (index) => {
    if (answers[index] !== null) {

 return isMarkedForReview[index] ? 'warning' : 'success'; // Answered and marked for review vs. answered
    } else {
      return isMarkedForReview[index] ? 'info' : 'light'; // Not answered but marked vs. not answered
    }
  };

  return (
    <Container className="my-4">
      <Card>
        <Card.Header>
          <h5>Math Test</h5>
        </Card.Header>
        <Card.Body>
          {/* Question Section */}
          <Row className="mb-4">
            <Col>
              <h5>Question No. {currentQuestionIndex + 1}</h5>
```

```
            <p>{currentQuestion.question}</p>

            <Form>

              {currentQuestion.options.map((option, index) => (

                <Form.Check

                  key={index}

                  type="radio"

                  label={option}

                  name={`question${currentQuestionIndex}`}

                  id={`option${index}`}


          checked={selectedOption === `option${index}`}

                  onChange={handleOptionChange}

                />

              ))}

            </Form>

          </Col>

        </Row>


        {/* Action Buttons */}

        <Row className="mb-4">

          <Col>

            <Button

              variant={isMarkedForReview[currentQuestionIndex] ? "warning"
: "primary"}

              className="me-2"

              onClick={handleMarkForReview}

            >
```

```
                {isMarkedForReview[currentQuestionIndex] ? "Unmark for
Review" : "Mark for Review"}

              </Button>

              <Button variant="danger" className="me-2"
onClick={handleClearResponse}>Clear Response</Button>

              <Button variant="success" onClick={handleSaveAndNext}>Save &
Next</Button>

            </Col>

          </Row>


          {/* Timer and Question Palette */}

          <Row>


            <Col md={6} className="mb-4">

              <div className="border p-3">

                <h6>Time Remaining</h6>

                <div className="text-center">

                  <h5>{formatTime(timer)}</h5>

                </div>

              </div>

            </Col>

            <Col md={6} className="mb-4">

              <div className="border p-3">

                <h6>Question Palette</h6>

                <div className="d-flex flex-wrap">

                  {questions.map((_, index) => (

                    <Button

                      key={index}
```

```
                    variant={getButtonVariant(index)}
                    className="m-1"
                    onClick={() => {
                      setCurrentQuestionIndex(index);
                      setSelectedOption(answers[index]);
                    }}
                  >
                    {index + 1}
                  </Button>
                ))}
              </div>
              <div className="mt-3">

                <Badge pill bg="success" className="me-2">Answered</Badge>
                 <Badge pill bg="danger" className="me-2">Not
Answered</Badge>
                 <Badge pill bg="warning" className="me-2">Marked</Badge>
                 <Badge pill bg="info" className="me-2">Answered & Marked
for Review</Badge>
              </div>
            </div>
          </Col>
        </Row>
      </Card.Body>
    </Card>
  </Container>
 );
};
```

```
export default TestM;
```

Test List :

```
import React from 'react'
import { Button, Card, Row, Col } from 'react-bootstrap'


export default function TestListItem(props) {
    return (
        <Card className='my-3 rounded-end'>
            <Card.Body className='align-middle'>
                <Row>
                    <Col>
                        <p>{props.test.testName}</p>
                    </Col>
                    <Col>
                        <Button className='float-right' variant='primary'>
Register</Button>
                    </Col>
                </Row>


            </Card.Body>
        </Card>
    )
}
```

Admin Dashboard :

```
    import React from 'react';
    import { Button, Container, Row, Card } from 'react-bootstrap';
    import createTest from '../Resources/create-test.png';
    import testlist from '../Resources/list-of-test.png';
    import registration from '../Resources/registration.png';
    import api from '../service/api'; // Ensure api.js is correctly
imported


    export default function AdminDashboard(props) {
  const styleImg = {
    width: "18rem",
    height: "18rem",
    padding: "3rem"
  };


  const createNewTest = async (e) => {
    e.preventDefault();


      try {
     const response = await api.post('/create-test', {
       testName: 'Sample Test',
       questions: [
         { question: 'Sample Question 1', options: ['A', 'B', 'C', 'D'],
correctOption: 0 },
       ],
     });
     console.log(response.data);
```

```
    } catch (error) {
      console.error('Error creating test:', error);
    }
  }

  const createdTests = async () => {
    try {
      const response = await api.get('/created-tests');
      console.log(response.data);
    } catch (error) {
      console.error('Error fetching created tests:', error);
    }
  }

  const registeredStudents = async () => {
    try {
      const response = await api.get('/registered-students');
      console.log(response.data);
    } catch (error) {
      console.error('Error fetching registered students:', error);
    }
  }
  return (
    <Container className='my-5 px-5'>
      <Row className='mx-1'>
        <Card className='shadow-lg text-center mx-4' style={{ width: '18rem'
}}>
          <Card.Img variant="top" src={createTest} style={styleImg} />
```

```
          <Card.Body>
            <Card.Title>Create Test</Card.Title>
            <Card.Text>
              Create a new MCQ test for your students.
            </Card.Text>
            <Button variant="primary" onClick={createNewTest}>Create
Test</Button>
          </Card.Body>
        </Card>

        <Card className='shadow-lg text-center mx-4' style={{ width: '18rem'
}}>
          <Card.Img variant="top" src={testlist} style={styleImg} />
          <Card.Body>
            <Card.Title>Created Tests</Card.Title>
            <Card.Text>
              View or edit tests that you have created.
            </Card.Text>
            <Button variant="primary" onClick={createdTests}>Available
Tests</Button>
          </Card.Body>
        </Card>

        <Card className='shadow-lg text-center mx-4' style={{ width: '18rem'
}}>
          <Card.Img variant="top" src={registration} style={styleImg} />
          <Card.Body>
            <Card.Title>Registered Students</Card.Title>
```

```
      <Card.Text>
            List of students registered for tests.
          </Card.Text>
          <Button variant="secondary"
onClick={registeredStudents}>Unavailable</Button>
        </Card.Body>
      </Card>
    </Row>
  </Container>
 );
}
```

Admin Login :

```
    import React, { useState } from 'react';
    import api from '../service/api'; // Ensure api.js is correctly
imported

    export default function AdminLogin(props) {
  const [adminName, setAdminName] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");

  const submit = async (e) => {
    e.preventDefault();
    if (!adminName || !password) {
      setError("Please fill in all fields");
    } else {
      try {
```

```
                const response = await api.post('/login', { adminName,
password });

                if (response.data.success) {

                    props.loginAdmin(adminName, password);

                    setAdminName("");

                    setPassword("");

                    setError("");

                } else {

                    setError("Invalid credentials");

                }

            } catch (error) {

                setError("An error occurred");

            }

        }

    }


    return (

        <div className="container">

            <div className="card o-hidden border-0 shadow-lg my-5" style={{
backgroundColor: 'rgba(255, 255, 255, 0.8)' }}>

                <div className="card-body p-0">

                    <div className="row">

                        <div className="col-lg-7 mx-auto" align="center">

                            <div className="p-5">

                                <div className="text-center">

                                    <h1 className="h4 text-gray-900
mb-4">Admin Login</h1>

                                </div>
```

```
                                {error && <p style={{ color: 'red'
}}>{error}</p>}

                                <form className="user" onSubmit={submit}
method="post" autoComplete="off">

                                    <div className="form-group">

                                        <input type="text" value={adminName}
onChange={(e) => setAdminName(e.target.value)} className="form-control
form-control-user" id="adminName"

                                            placeholder="Admin Name" />

                                    </div>

                                    <div className="form-group">

                                        <input type="password"
value={password} onChange={(e) => setPassword(e.target.value)}
className="form-control form-control-user"

                                            id="password"
placeholder="Password" />

                                    </div>

                                    <div className="form-group">

                                        <button type="submit" className="btn
btn-primary btn-user btn-block">Login</button>

                                    </div>

                                </form>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        )

}
```

Header :

```
    import React from 'react';
    import PropTypes from 'prop-types';
    import { Link } from "react-router-dom";
    import { Row, Col } from 'react-bootstrap';


    export default function Header(props) {


  const logout = (e) => {
    e.preventDefault();
    props.logout();
  }


  return (
    <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
      <Link className="navbar-brand"
        to={props.isLoggedIn ? (props.isAdmin ? "/admin-dashboard" :
"/student-dashboard") : "/"}>
        <h3>Online Exam Portal</h3>
      </Link>
      <ul className="navbar-nav mr-auto">


      </ul>
      {props.isLoggedIn ?
        <Row className='vertical-center'>
          <Col className='text-center pt-1 text-white text-nowrap'>
            {props.isAdmin ? "Admin" : props.studentName}
```

```
          </Col>

          <Col>

            <form className="form-inline my-2 my-lg-0" onSubmit={logout}>

              <button className="btn btn-primary my-2 my-sm-0"
type="submit">

                Logout

              </button>

            </form>

          </Col>

        </Row> : ""}

    </nav>

  )

}

Header.defaultProps = {

  title: "Your Title Here",

  searchBar: true

}




    Header.propTypes = {

  title: PropTypes.string,

  searchBar: PropTypes.bool.isRequired

}
```

Main :

```
import React from 'react';

import { Card, Container, Row, Col, Button } from 'react-bootstrap';

import { Link } from 'react-router-dom';
```

```
export const Main = () => {

    return (

        <Container className='text-center my-5' >

            <Card  style={{backgroundColor: 'rgba(255, 255, 255, 0.7)'}}>

                <Card.Header className='text-center'style={{backgroundColor:
'rgba(255, 255, 255, 0.9)'}}>

                    <h2>Welcome to the Exam Portal!</h2>

                </Card.Header>

                <Card.Body className='text-center' style={{backgroundColor:
'rgba(255, 255, 255, 0.6)'}}>

                    <Row className='p-5'>

                        <Col></Col>

                        <Col>

                            <Link to="/student-login"><Button
variant="primary" size="lg">Student Login</Button></Link>

                        </Col>


                        <Col>

                            <Link to="/admin-login"><Button
variant="primary" size="lg">Admin Login</Button></Link>

                        </Col>


                        <Col></Col>

                    </Row>

                    <Row className='text-center'>

                        <Col>

                        <h5 className="text-center">New Student?<Link
to="/register-student"> Register now.</Link></h5>
```

```
                        </Col>


                    </Row>
                </Card.Body>
            </Card>
        </Container>
    )
}
```

Register Students :

```
import { Link } from "react-router-dom";
import react , {useState} from "react";
export default function RegisterStudent(props) {


    const [student, setStudent] = useState({
        studentName: "",
        email: "",
        mobNo: null,
        pass: "",
        address: {
            city: "",
            state: "",
            country: "",
            zip: null
        }
    });


    const submit = (e) => {
```

```
    console.log(student);
    props.registerStudent(student);
    setStudent({studentName: "",
    email: "",
    mobNo: null,
    pass: "",
    address: {
        city: "",
        state: "",
        country: "",
        zip: null
    }});
    e.preventDefault();
}

return (
    <div className="container">
        <div className="card o-hidden border-0 shadow-lg my-5"
style={{backgroundColor: 'rgba(255, 255, 255, 0.9)'}}>
            <div className="card-body p-0">
                <div className="row">
                    <div className="col-lg-5 d-none d-lg-block
bg-register-image" ></div>
                    <div className="col-lg-7" >
                        <div className="p-5" >
                            <div className="text-center">
                                <h1 className="h4 text-gray-900
mb-4">Student Registration</h1>
```

```
                        </div>


  <form className="user" onSubmit={submit} method="post"
autoComplete="off">
                                <div className="form-group">
                                    <input
                                        type="text"
                                        value={student.studentName}
                                        onChange={(e) => setStudent({
...student, studentName: e.target.value })}
                                        className="form-control
form-control-user"
                                        id="studentName"
                                        placeholder="Student Name" />
                                </div>
                                <div className="form-group">
                                    <input
                                        type="email"
                                        value={student.email}
                                        onChange={(e) => setStudent({
...student, email: e.target.value })}
                                        className="form-control
form-control-user"
                                        id="exampleInputEmail"
                                        placeholder="Email Address" />
                                </div>
                                <div className="form-group row">
                                    <div className="col-sm-6 mb-3
mb-sm-0">
                                        <input
```

```
                                        type="text"
                                        value={student.mobNo}
                                        onChange={(e) => setStudent({
...student, mobNo: e.target.value })}
                                        className="form-control
form-control-user"
                                        id="mobileNo"
                                        placeholder="Mobile no." />

    </div>
                                <div className="col-sm-6">
                                    <input
                                        type="password"
                                        value={student.pass}
                                        onChange={(e) => setStudent({
...student, pass: e.target.value })}
                                        className="form-control
form-control-user"
                                        id="password"
                                        placeholder="Password" />
                                </div>
                            </div>
                            <div className="form-group row">
                                <div className="col-sm-6 mb-3
mb-sm-0">
                                    <input
                                        type="text"
                                        value={student.address.city}
                                        onChange={(e) => setStudent({
...student, address: { ...student.address, city: e.target.value } })}
```

```
                                        className="form-control
form-control-user"

                                        id="city"

                                        placeholder="City" />

                                </div>

                                <div className="col-sm-6">

                                    <input

                                        type="text"

                                        value={student.address.state}

                                        onChange={(e) => setStudent({
...student, address: { ...student.address, state: e.target.value } })}

                                        className="form-control
form-control-user"

                                        id="state"


            placeholder="State" />

                                </div>

                            </div>

                            <div className="form-group row">

                                <div className="col-sm-6 mb-3
mb-sm-0">

                                    <input

                                        type="text"


value={student.address.country}

                                        onChange={(e) => setStudent({
...student, address: { ...student.address, country: e.target.value } })}

                                        className="form-control
form-control-user"

                                        id="country"
```

```
                                        placeholder="Country" />
                                </div>
                                <div className="col-sm-6">
                                    <input
                                        type="text"
                                        value={student.address.zip}
                                        onChange={(e) => setStudent({
...student, address: { ...student.address, zip: e.target.value } })}
                                        className="form-control
form-control-user"
                                        id="zip"
                                        placeholder="Zipcode" />
                                </div>
                            </div>
                            <div className="form-group row">
                                <div className="col-sm-6 mb-3
mb-sm-0">
                                    <button type="submit"
className="btn btn-primary btn-user btn-block">Register</button>
                                </div>
                                <div className="col-sm-6 mb-3
mb-sm-0">

      <button type="reset" className="btn btn-primary btn-user
btn-block">Reset</button>
                                </div>
                            </div>
                        </form>
                        <hr />
                        <div className="text-center">
```

```
                                    <Link className="medium"
    to="/student-login">Already have an account? Login!</Link>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>


        </div>

    )

}Student Dashboard
```

Student Dashboard :

```
    import React from 'react';

    import Card from 'react-bootstrap/Card';

    import { Button, Container, Row } from 'react-bootstrap';

    import userprofie from '../Resources/user-profile.png';

    import testlist from '../Resources/list-of-test.png';

    import performance from '../Resources/performance.png';

    import api from '../service/api'; // Import the API service




    export default function StudentDashboard(props) {


  const styleImg = {

    width: "18rem",
```

```
  height: "18rem",
  padding: "3rem"
};

const openProfile = async (e) => {
  e.preventDefault();
  try {
    const response = await api.get('/profile'); // Replace with actual
  endpoint
    if (response.data.success) {
      // Handle the profile data
      console.log(response.data.profile);
    } else {
      console.error('Failed to fetch profile');
    }
  } catch (error) {
    console.error('Error fetching profile:', error);
  }
}

const availableTest = async (e) => {
  e.preventDefault();
  try {
    const response = await api.get('/available-tests'); // Replace with
  actual endpoint
    if (response.data.success) {
      // Handle the available tests data
```

```
console.log(response.data.tests);

    } else {

      console.error('Failed to fetch available tests');

    }

  } catch (error) {

    console.error('Error fetching available tests:', error);

  }

}


const trackRecord = (e) => {

  e.preventDefault();

  // Handle track record functionality here

}


return (

  <>

    <Container className='my-5 px-5'>

      <Row className='mx-1'>

        <Card className='shadow-lg text-center mx-4' style={{ width: '18rem'
  }}>

          <Card.Img variant="top" src={userprofie} style={styleImg} />

          <Card.Body>

            <Card.Title>Your Profile</Card.Title>

            <Card.Text>

              View or edit details in your student profile.

            </Card.Text>

            <Button variant="primary" onClick={openProfile}>Open
  Profile</Button>
```

```
        </Card.Body>
      </Card>



<Card className='shadow-lg text-center mx-4' style={{ width: '18rem' }}>
        <Card.Img variant="top" src={testlist} style={styleImg} />
        <Card.Body>
          <Card.Title>View Available Tests</Card.Title>
          <Card.Text>
            Check out the latest list of available tests for you.
          </Card.Text>
          <Button variant="primary" onClick={availableTest}>Available
Tests</Button>
        </Card.Body>
      </Card>


      <Card className='shadow-lg text-center mx-4' style={{ width: '18rem'
}}>
        <Card.Img variant="top" src={performance} style={styleImg} />
        <Card.Body>
          <Card.Title>Track Records</Card.Title>
          <Card.Text>
            Under Maintenance. Will be available soon.
          </Card.Text>
          <Button variant="secondary"
onClick={trackRecord}>Unavailable</Button>
        </Card.Body>
      </Card>
```

```
        </Row>
      </Container>
    </>
  )
}
```

Student Login :

```
import React, { useState } from 'react';
import api from '../service/api'; // Ensure the path is correct
export default function StudentLogin(props) {
const [email, setEmail] = useState("");
const [password, setPassword] = useState("");
const submit = async (e) => {
    e.preventDefault();
    if (!email || !password) {
    } else {
        try {
            const response = await api.post('/login', { email, password
});
            if (response.data.success) {
                props.loginStudent(response.data.student); // Assuming
this updates the parent component state
            } else {
                console.error('Login failed:', response.data.message);
            }
        } catch (error) {
            console.error('Error during login:', error);
```

```
        }

        setEmail("");

        setPassword("");

    }

};

return (

    <div className="container">

        <div className="card o-hidden border-0 shadow-lg my-5" style={{
backgroundColor: 'rgba(255, 255, 255, 0.8)' }}>

            <div className="card-body p-0">

                <div className="row">

                    <div className="col-lg-7 mx-auto" align="center">

                        <div className="p-5">

                            <div className="text-center">

                                <h1 className="h4 text-gray-900
mb-4">Student Login</h1>

                            </div>

                            <form className="user" onSubmit={submit}
method="post" autoComplete="off">

                                <div className="form-group">

                                    <input type="email" value={email}
onChange={(e) => setEmail(e.target.value)} className="form-control
form-control-user" id="email"

                                        placeholder="Email Address" />

                                </div>

                                <div className="form-group">

                                    <input type="password"
value={password} onChange={(e) => setPassword(e.target.value)}
className="form-control form-control-user"
```

```
                                        id="password"
placeholder="Password" />

                                    </div>

                                    <div className="form-group">

                                        <button type="submit" className="btn
btn-primary btn-user btn-block">Login</button>

                                    </div>

                                </form>

                                <hr />

                                <div className="text-center">

                                    <p className="medium">New Student?<a
href="/register-student"> Register now.</a></p>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </div> )}
```

```
                        // View : Students ---Code

              import React from 'react';
import { Container, Row, Card, Button } from 'react-bootstrap';
import { useParams } from 'react-router-dom';


export default function ViewStudent() {
  const { studentId } = useParams(); // Get the student ID from the URL


  // Dummy student data for demonstration
  const student = {
    id: studentId,
    name: 'John Doe',
    email: 'johndoe@example.com',
    registrationDate: '2023-01-15',
  };


  return (
    <Container className='my-5'>
      <Row className='justify-content-center'>
        <Card className='shadow-lg text-center' style={{ width: '30rem' }}>
          <Card.Body>
            <Card.Title>Student Details</Card.Title>
            <Card.Text><strong>ID:</strong> {student.id}</Card.Text>
            <Card.Text><strong>Name:</strong> {student.name}</Card.Text>
            <Card.Text><strong>Email:</strong> {student.email}</Card.Text>
            <Card.Text><strong>Registration Date:</strong>
    {student.registrationDate}</Card.Text>
```

```
            <Button variant="primary" onClick={() =>
    window.history.back()}>Back</Button>

          </Card.Body>

        </Card>

      </Row>

    </Container>

  );

}
```

App.CSS FILE :

```
.App {  text-align: center;

}


.App-logo {

  height: 40vmin;

  pointer-events: none;

}




.bg-register-image {

  background: url(./Resources/register-img.jpg);

  background-position: center;

  background-size: cover;

}

.myImg {

  z-index: -5;

  background-image: url(./Resources/bg-img.jpg);

  height: 91vh;
```

```
  background-size: cover;
  display: flex;
  align-items: center;
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;}
```

App.Js:

```
import './App.css';
import { Main } from './Components/Main';
import {
  Routes,
  Route,
  useNavigate
} from "react-router-dom";
```

```
import StudentLogin from './Components/StudentLogin';
import AdminLogin from './Components/AdminLogin';
import RegisterStudent from './Components/RegisterStudent';
import Header from './Components/Header';
import StudentDashboard from './Components/StudentDashboard';
import AdminDashboard from './Components/AdminDashboard';
import { useState } from 'react';
import StudentProfile from './Components/StudentDashboard/StudentProfile';
import AvailableTest from './Components/StudentDashboard/AvailableTest';
import TrackRecords from './Components/StudentDashboard/TrackRecords';
import CreateTest from './Components/AdminDashboard/CreateTest';


function App() {


  const navigate = useNavigate();
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [isAdmin, setisAdmin] = useState(false);
  const [studentId, setstudentId] = useState(null);
  let [student, setStudent] = useState({
    studentName: "",
    email: "",
    mobNo: null,
    pass: "",
    address: {
      city: "",
      state: "",
```

```
    country: "",
    zip: null
  }
});
let [studentTestList, setstudentTestList] = useState(null);
const [adminId, setadminId] = useState(null);
const [studentName, setstudentName] = useState(null);


const logout = () => {
  setIsLoggedIn(false);
  setisAdmin(false);
  navigate("/");


}


const loginStudent = (email, pass) => {


  const requestOptions = {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      email: email,
      pass: pass
    })
  };
```

```
    fetch('/portal/login', requestOptions)
      .then(response => response.json())
      .then(data => {
        if (data.studentId) {
          setIsLoggedIn(true);
          setisAdmin(false);
          setstudentId(data.studentId);
          setstudentName(data.studentName);
          navigate("/student-dashboard");
        } else {
          setIsLoggedIn(false);
          setisAdmin(false);
          alert("Username/Password not correct");
        }
      });



}

const loginAdmin = (name, pass) => {
  const requestOptions = {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      adminName: name,
```

```
        pass: pass
      })
    };
    fetch('/portal/admin-login', requestOptions)
      .then(response => response.json())
      .then(data => {
        console.log(data);
        if (data.adminId) {
          setIsLoggedIn(true);
          setisAdmin(true);
          setadminId(data.adminId);
          navigate("/admin-dashboard");
        } else {
          setIsLoggedIn(false);
          setisAdmin(false);
          alert("Adminname/Password not correct");
        }
      });
  }

  const registerStudent = (student) => {
    const requestOptions = {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        ...student
```

```
      })
    };
    fetch('/portal/create-student', requestOptions)
      .then(response => response.json())
      .then(data => {
        if (data > 0) {
          alert("Registration Successful!!");
          navigate("/");
        } else {
          alert("Please retry, registration unsuccessful.");
        }
      });
  }

  const openProfile = () => {
    const requestOptions = {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        studentId: studentId
      })
    };
    fetch('/portal/student-details', requestOptions)
      .then(response => response.json())
      .then(data => {
        if (data.studentId) {
```

```
          setStudent({
            ...student,
            studentName: data.studentName,
            studentId: data.studentId,
            email: data.email,
            mobNo: data.mobNo,
            pass: data.pass,
            address: {
              ...student.address,
              addrId: data.address.addrId,
              city: data.address.city,
              state: data.address.state,
              country: data.address.country,
              zip: data.address.zip
            }
          });
          navigate("/student-dashboard/student-profile");
        } else {
          alert("We're facing some network issues!");
        }
      });
  }


  const saveStudentDetails = (student) => {
    const requestOptions = {
      method: 'PUT',
```

```
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        ...student
      })
    };
    fetch('/portal/edit-student-details', requestOptions)
      .then(response => response.json())
      .then(data => {
        if (data.studentId === studentId) {
          alert("Details saved successfully!");
          setstudentName(data.studentName);
          openProfile();
        } else {
          alert("Please retry, failed to save changes.");
        }
      });
  }

  const openDashboard = () => {
    navigate('/student-dashboard');
  }

  const openAdminDashboard = () => {
    navigate('/admin-dashboard');
  }
```

```
const availableTest = () => {
  fetch('/portal/all-test-list')
    .then(response => response.json())
    .then(data => {
      if (data.testDTOList) {
        setstudentTestList({ ...data.testDTOList })
        navigate("/student-dashboard/available-test");
      } else {
        alert("No tests available right now.");
      }
    });
}

const createTest = () => {
  navigate('/admin-dashboard/create-new-test');
}

const saveTest = (test) => {
  const requestOptions = {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      ...test
    })
  };
```

```
        fetch('/portal/create-test', requestOptions)
          .then(response => response.json())
          .then(data => {
            if (data > 0) {
              alert("Test Created Successful!!");
              navigate("/admin-dashboard");
            } else {
              alert("Please retry, test not saved.");
            }
          });
      }

      return (
        <>
          <Header title="Online Exam Portal" isLoggedIn={isLoggedIn}
isAdmin={isAdmin} logout={logout} studentName={studentName} />
          <div className="myImg">
          <Routes>
            <Route path='/' element={<Main />} />
            <Route path='/student-login' element={<StudentLogin
loginStudent={loginStudent} />} />
        <Route path='/admin-login' element={<AdminLogin loginAdmin={loginAdmin}
/>} />
            <Route path='/register-student' element={<RegisterStudent
registerStudent={registerStudent} />} />
            <Route path='/student-dashboard' element={<StudentDashboard
openProfile={openProfile} availableTest={availableTest} />} />
```

```
            <Route path='/admin-dashboard' element={<AdminDashboard
createTest={createTest} />} />

            <Route path='/admin-dashboard/create-new-test' element={<CreateTest
adminId={adminId} saveTest={saveTest} openAdminDashboard={openAdminDashboard}/>}
/>

            <Route

              path='/student-dashboard/student-profile'

              element={<StudentProfile

                student={student}

                saveStudentDetails={saveStudentDetails}

                openDashboard={openDashboard} />} />

            <Route

              path='/student-dashboard/available-test'

              element={<AvailableTest

                studentTestList={studentTestList}

                openDashboard={openDashboard} />} />

            <Route path='/student-dashboard/track-records'
element={<TrackRecords />} />

          </Routes>

          </div>

        </>

      );

    }


    export default App;
```

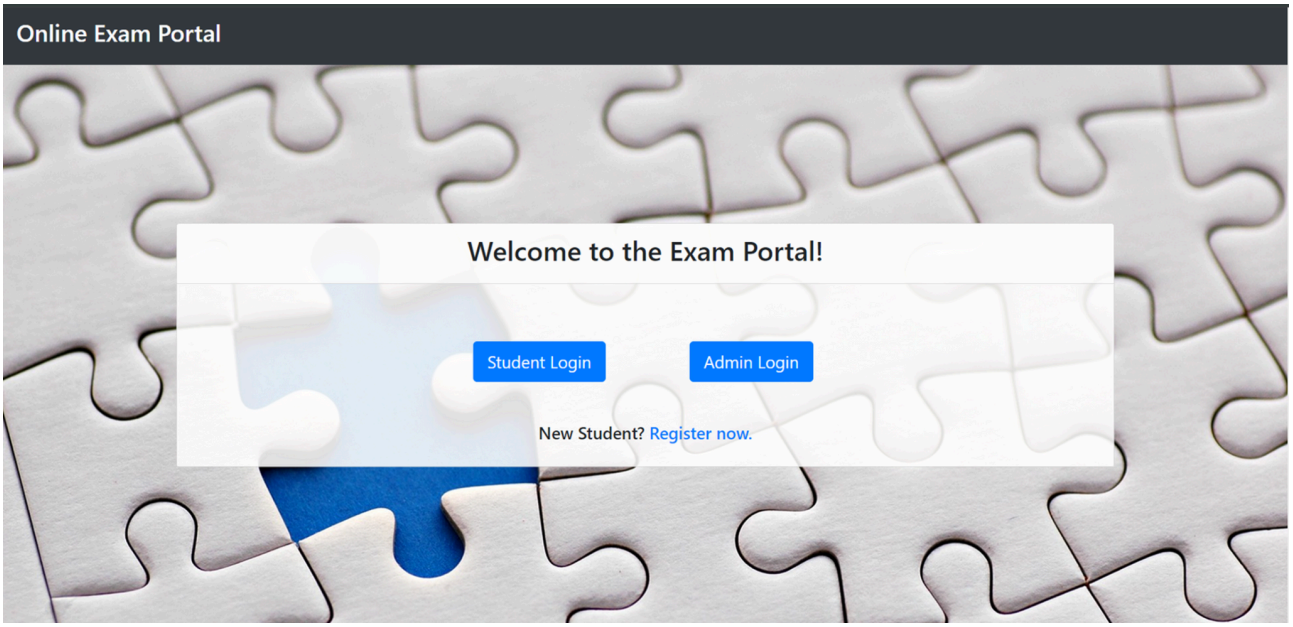
Index.js

```
    import React from 'react';
```

```
import {StrictMode} from 'react';
import {createRoot} from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';


const rootElement = document.getElementById('root');
const root = createRoot(rootElement);


root.render(
  <BrowserRouter>
  <StrictMode>
    <App />
  </StrictMode>
  </BrowserRouter>,
);


reportWebVitals();
```

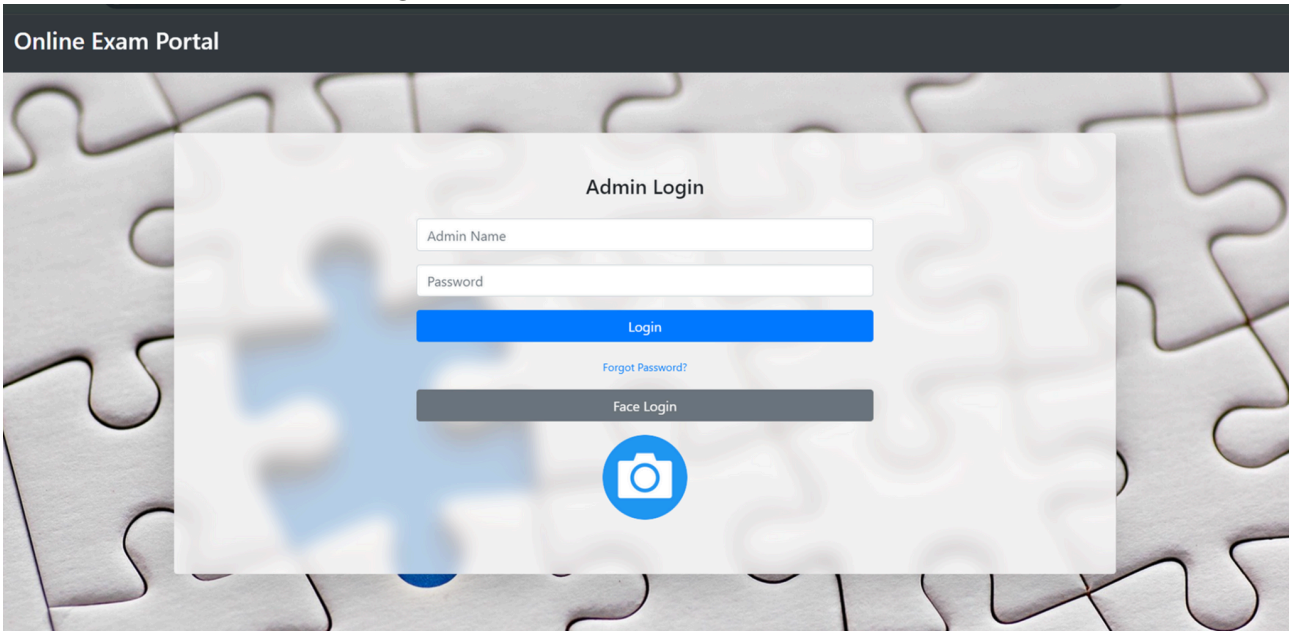
Home page:

It has a Header with the title “Online Exam Portal” & a main section with “Student Login” and “Admin Login” buttons. If there is any new student it will also show the link for “Registration”.

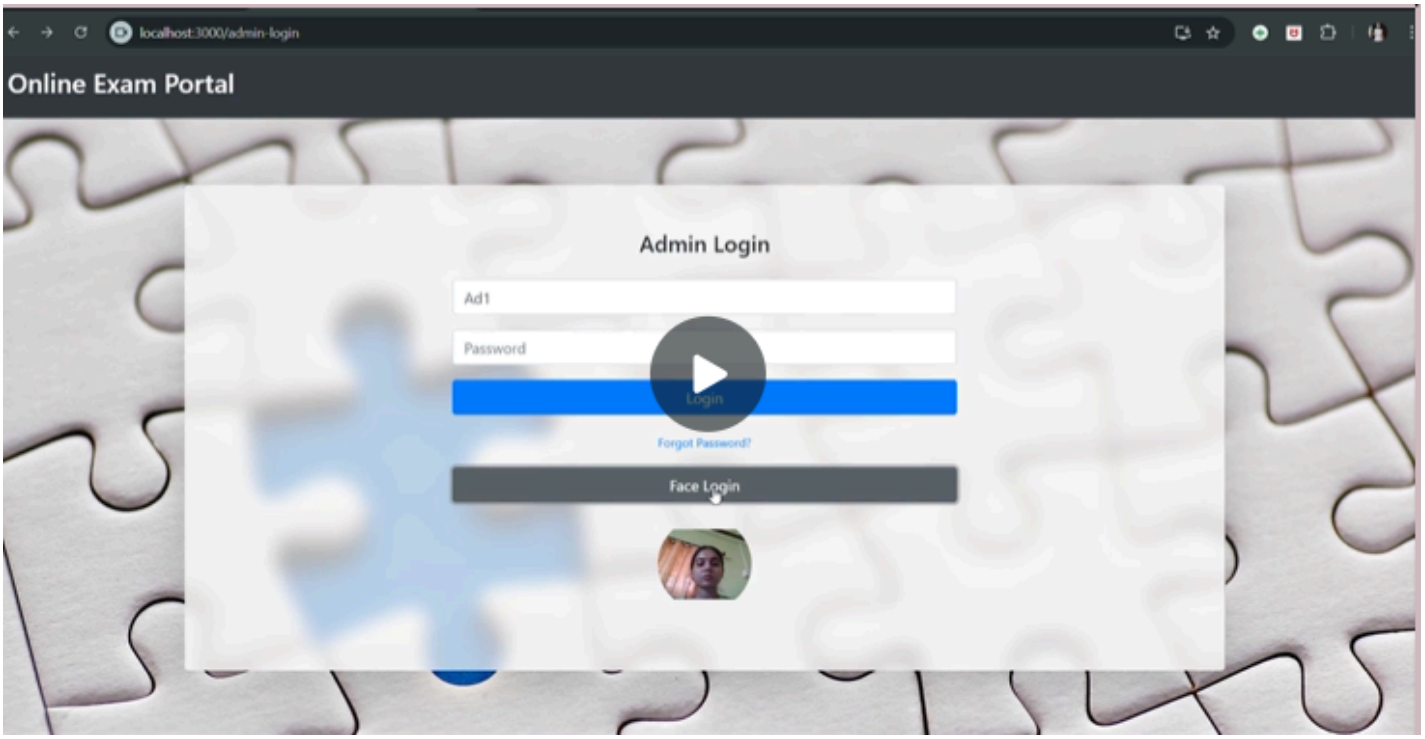


Admin Login Page :

An Admin can login through Id & Password. By any chance if the admin forgets the password he/she cant enter to the "Admin Dashboard". So, we have implemented a "Face Login" feature. Through the face login the admin can verify his/her face & can login. After successfully login it will show the Swal "Success" message. If the password is not given it will show a swal "Error" message.

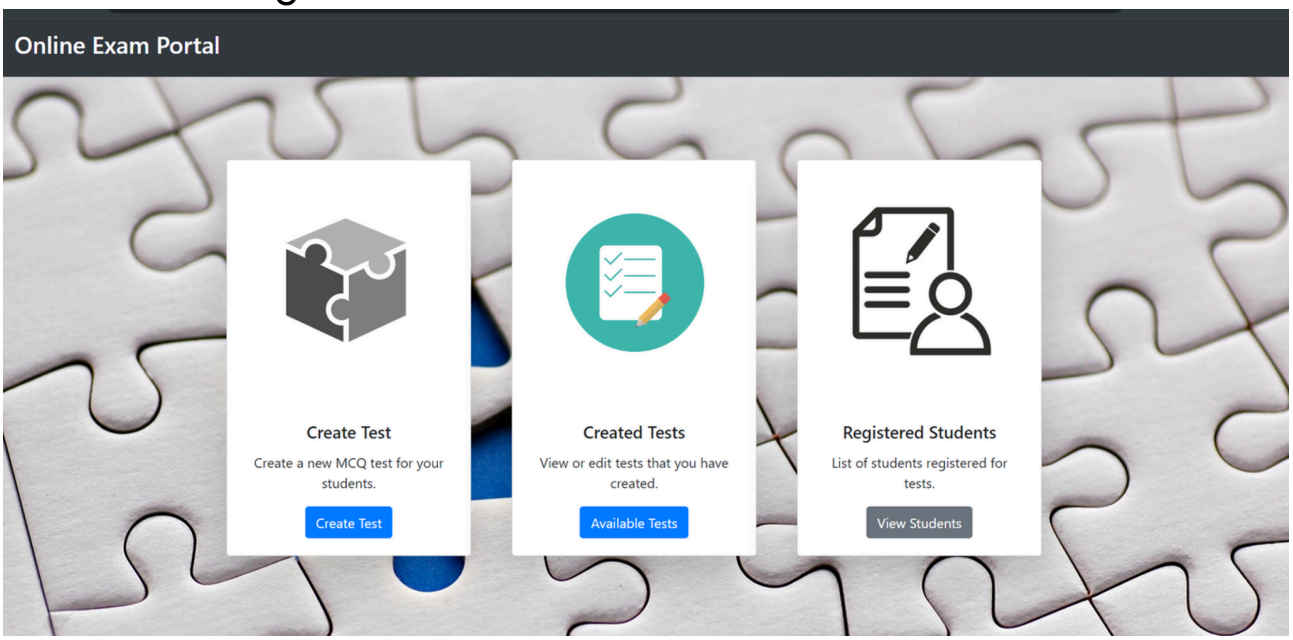


Face Login :



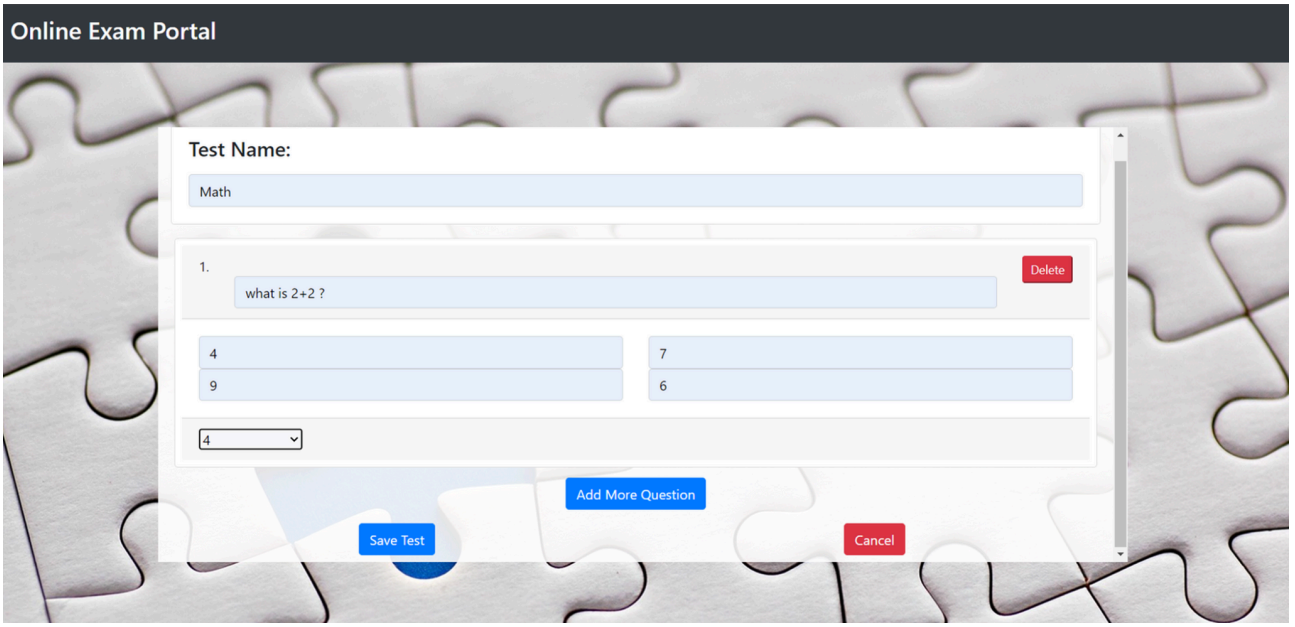
Admin Dashboard Page:

It has 3 individual pages, 1: “Create Test" ,where admin can add,edit & create tests; 2:"Created Tests", where admin can see the tests he/she has created or available right now and 3:"Registered Students", where admin can see the registered students



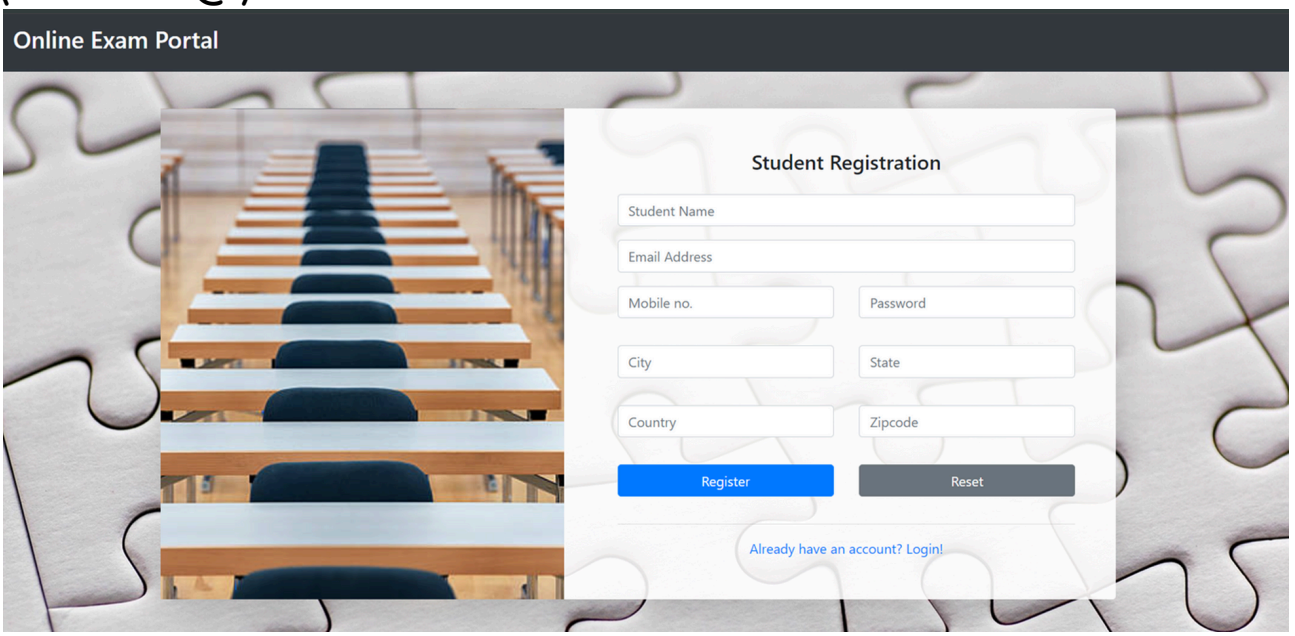
Create Test Page:

After clicking the “Create Tests" button the admin can create tests, add more questions, save the tests & if required then can cancel the tests.



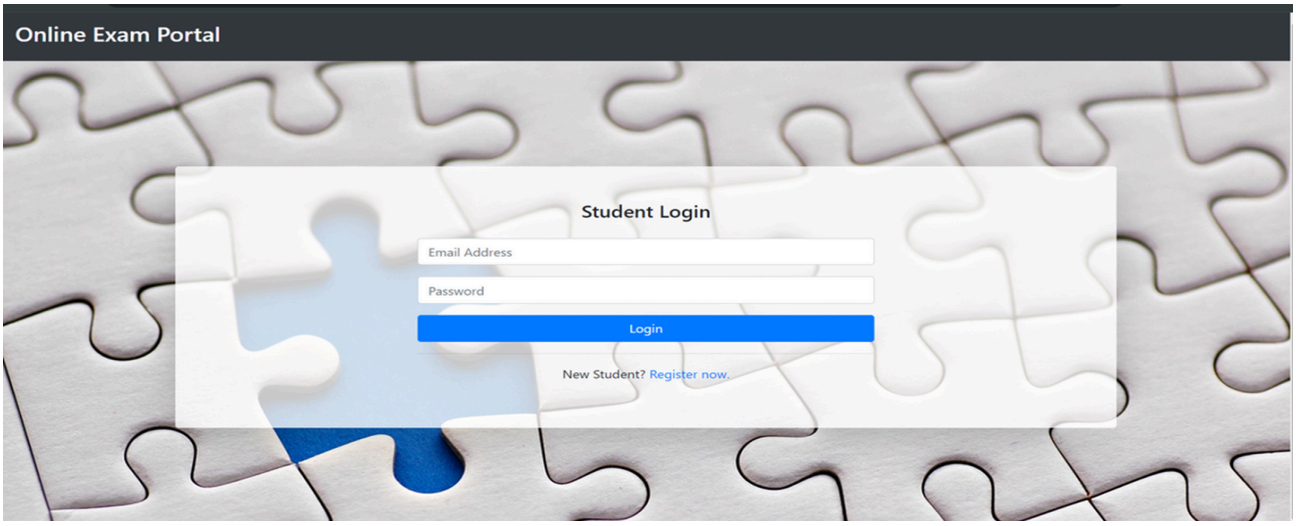
Student Registration Page:

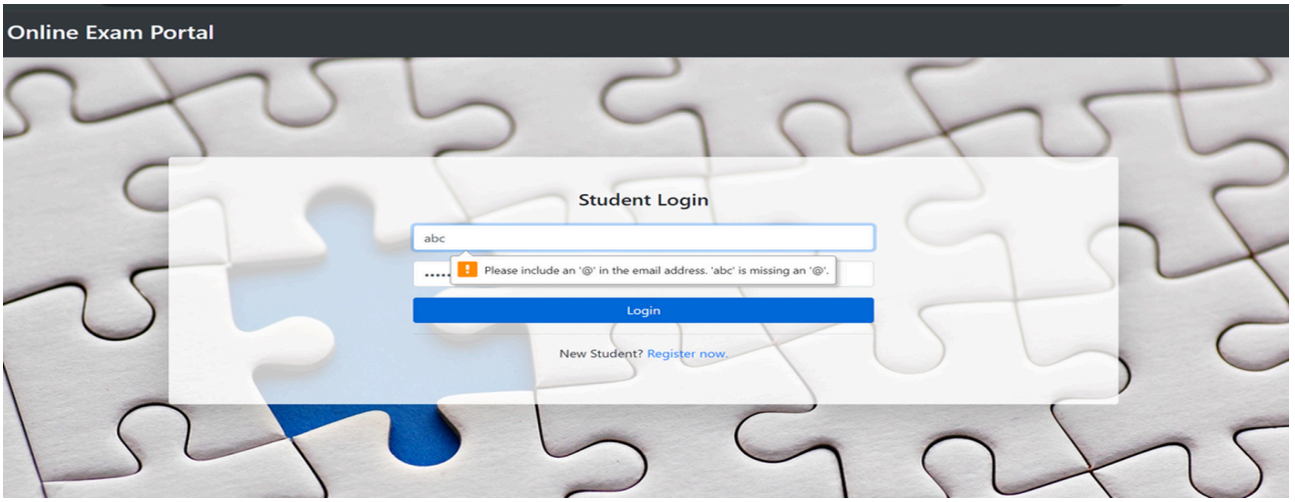
This page takes all necessary student details like name, password, email, phone no., city, country, pin code etc. There are some restrictions like the mobile/ phone no. should contain only digits & 10 digits exactly ;the password must contain at least 8 characters with at least 1 lowercase , 1 uppercase letter,1 symbol, 1 number; email must contain exact format (includes '@') etc.



Student Login Page:

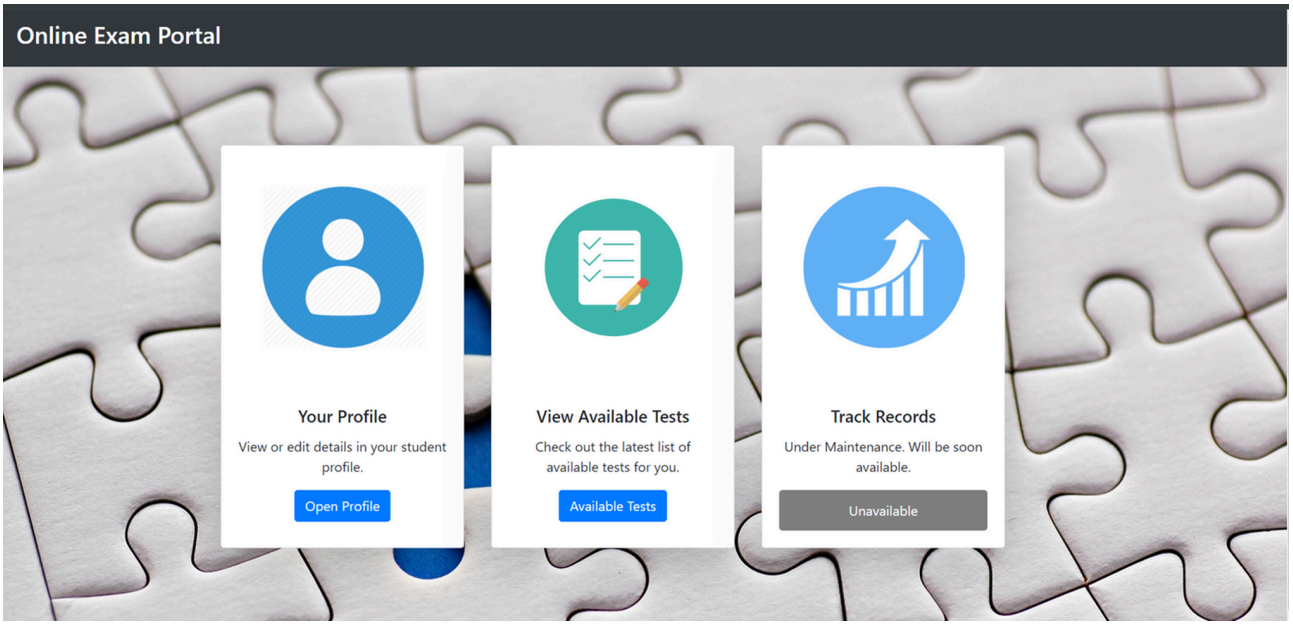
If the student is already registered then he/she can login through this page with mail id & password. This page also contains a link to the student registration page. If the student is not registered, then he/she will go to this page. There are some restrictions during login , the email id should be in its exact format.





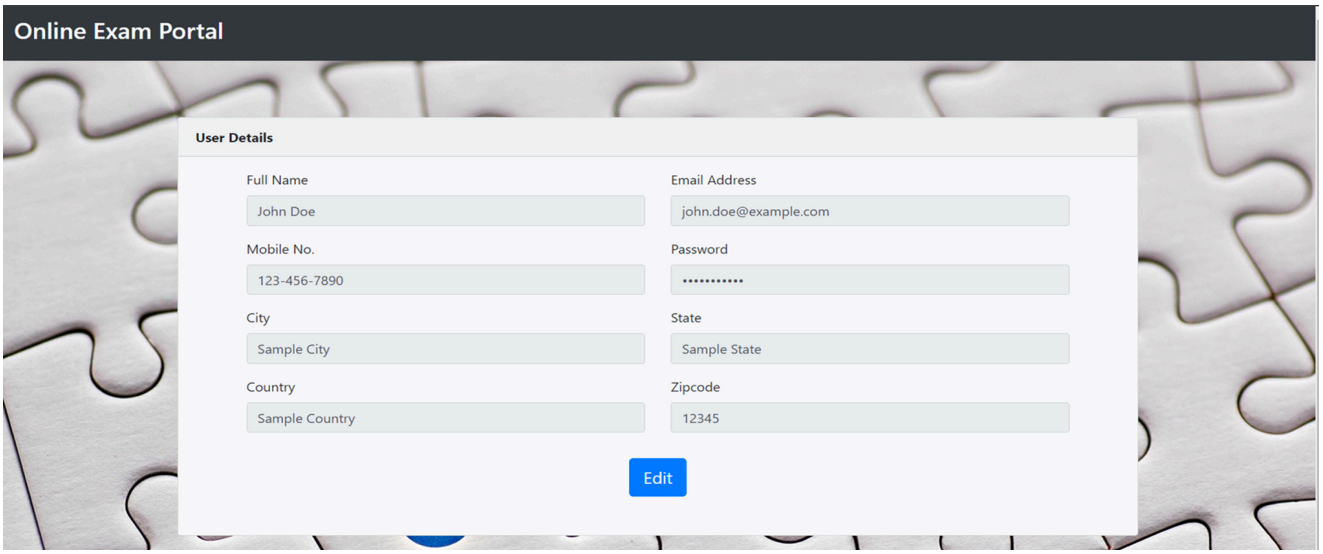
Student Dashboard Page:

After successful "login" or "registration" a student will arrive at this page. The student can see his/her profile with details, can view available tests & can track the records. For now "Track Records" is unavailable to a student, it will be available when the admin will give permission after an interval of time. If a student want to access the page it will show a block sign.



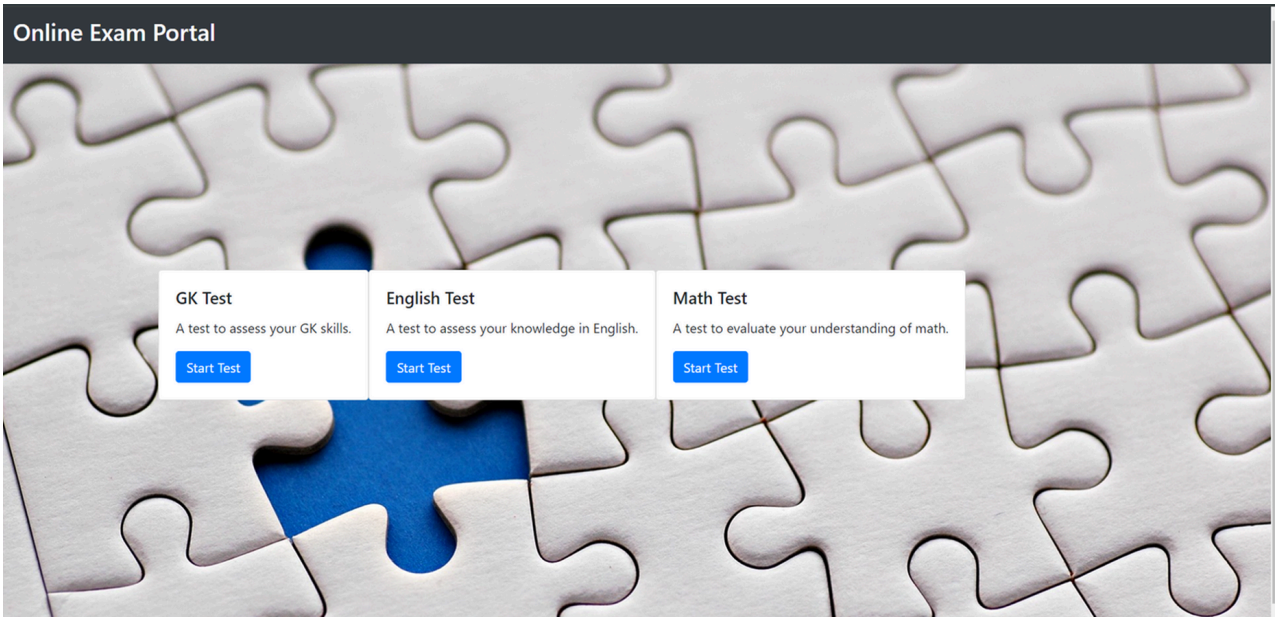
Student Profile Page:

On clicking "Open Profile" a student can see his/her detailed information. The student can also edit the details & can save or cancel changes.



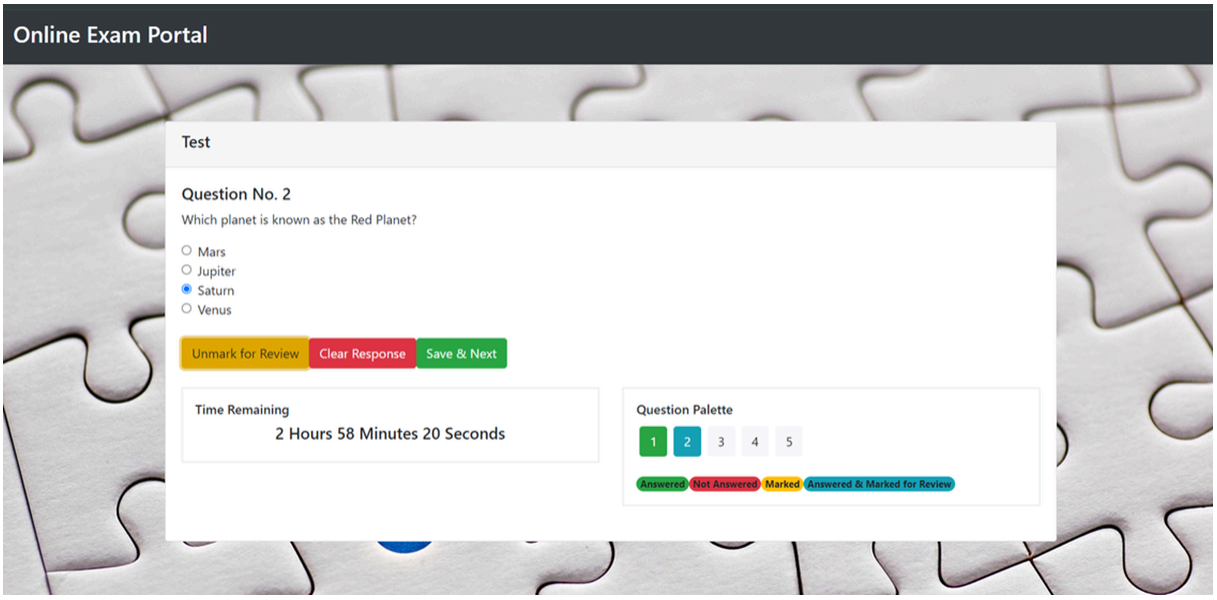
Available Test Page:

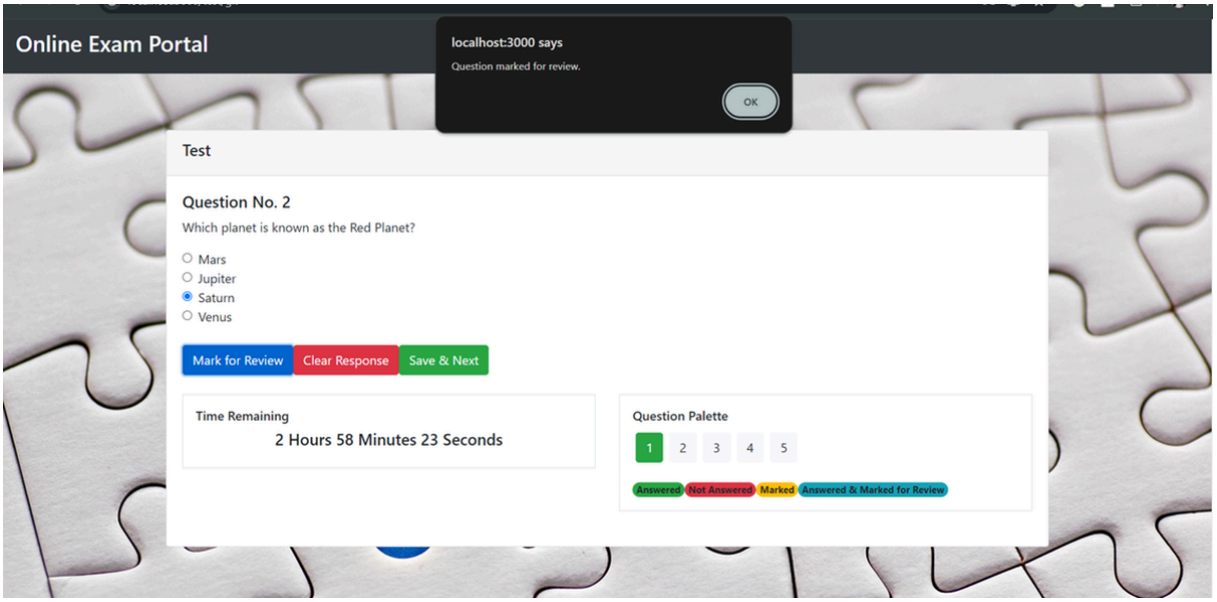
On clicking "Available Tests" a student can see the available tests. For now there are 3 tests available, 1:"GK Test", 2:"English Test", 3:"Math Test". On clicking "Start Test" the student can start the test. It will take him/her to another page where he/she can give the test.

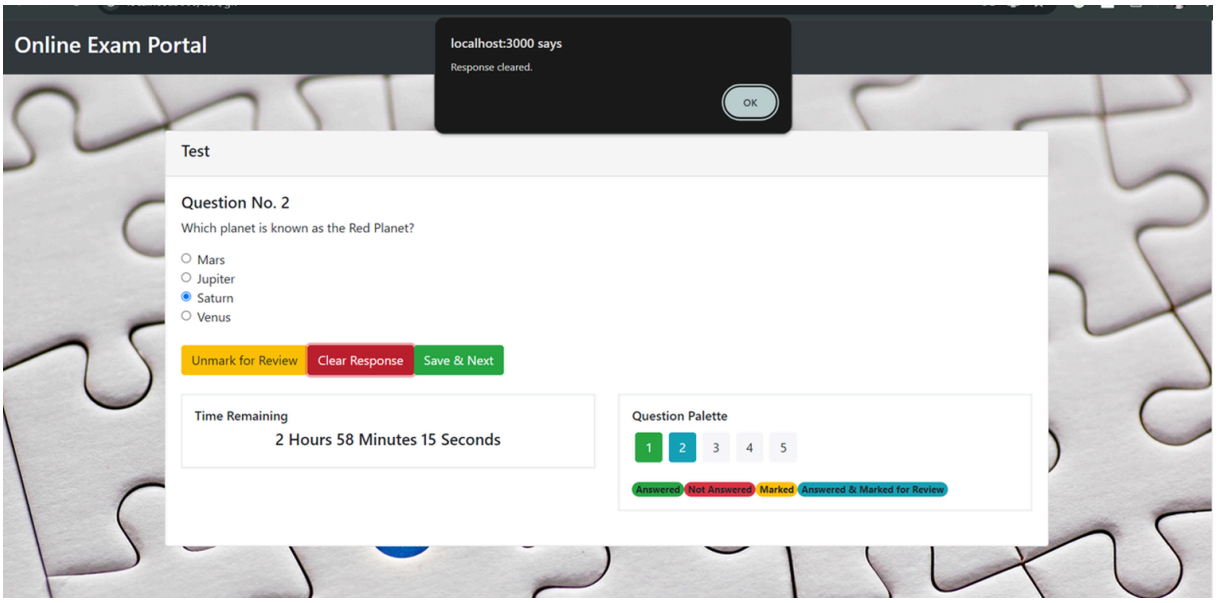


Test Pages:

A student can answer the MCQ & save it- in the question palette it will show "Green", i.e "Answered"; Clear the response- in the question palette it will show "Red", i.e "Not Answered"; Answer & mark it for review- in the question palette it will show "Blue", i.e "Answered & Marked for review"; Mark question for review- in the question palette it will show "Yellow", i.e " Marked ". All of this changes will be shown to a student within an "Alert box" .Below there is a "Timer" .


Online Exam Portal
Test
Question No. 2
Which planet is known as the Red Planet?
Mars
Jupiter
Saturn
Venus
Unmark for Review
Clear Response
Save & Next
Time Remaining
2 Hours 58 Minutes 20 Seconds
Question Palette
1
2
3
4
5
Answered
Not Answered
Marked
Answered & Marked for Review


Online Exam Portal
localhost:3000 says
Question marked for review.
OK
Test
Question No. 2
Which planet is known as the Red Planet?
Mars
Jupiter
Saturn
Venus
Mark for Review
Clear Response
Save & Next
Time Remaining
2 Hours 58 Minutes 23 Seconds
Question Palette
1
2
3
4
5
Answered
Not Answered
Marked
Answered & Marked for Review


Online Exam Portal
localhost:3000 says
Response cleared.
OK
Test
Question No. 2
Which planet is known as the Red Planet?
Mars
Jupiter
Saturn
Venus
Unmark for Review
Clear Response
Save & Next
Time Remaining
2 Hours 58 Minutes 15 Seconds
Question Palette
1
2
3
4
5
Answered
Not Answered
Marked
Answered & Marked for Review

CONCLUSION

The development of the Online Exam Panel marks a pivotal shift in the landscape of digital education. By harnessing the power of MERN stack technology, this platform has successfully addressed many of the limitations associated with traditional examination methods. The system's ability to facilitate seamless exam creation, management, and execution provides educators with the tools they need to deliver assessments efficiently, while students benefit from a streamlined and accessible examination process.

A key feature of this Online Exam Panel is the incorporation of face recognition technology for Admin login, which significantly enhances the security of the platform. This feature ensures that only authorised administrators can access and manage critical data, safeguarding the integrity of the examination process. Such robust security measures are essential in maintaining trust and credibility in digital assessments, particularly in an era where online education is becoming increasingly prevalent.

Moreover, the platform's user-friendly interface and intuitive design make it easy for both educators and students to navigate the system. The ability to manage exams, view results, and track performance in real-time offers users a comprehensive and engaging experience. The flexibility and scalability of the system further ensure that it can adapt to the evolving needs of educational institutions, accommodating a wide range of examination formats and requirements.

In conclusion, the Online Exam Panel offers a modern solution to the challenges of contemporary education, enhancing exam efficiency and security. Its adaptable platform is poised to support the evolving needs of digital education, playing a vital role in the future of academic assessments and fostering innovation in the field.

FUTURE SCOPE AND FURTHER ENHANCEMENTS

The future scope of the Online Exam Panel is expansive, with numerous opportunities for growth and innovation. As educational technology continues to evolve, the platform can be further enhanced to meet emerging needs and challenges. Potential enhancements include the integration of AI-driven analytics to provide personalised feedback and performance insights for students. This would enable more tailored learning experiences and help educators identify areas where students may need additional support.

Another area for future development is the inclusion of more advanced proctoring features, such as AI-based monitoring tools that can detect suspicious behaviours during exams. This would further enhance the integrity of the examination process, ensuring a fair testing environment for all students.

Additionally, the platform could be expanded to support a wider range of exam formats, including practical assessments, group exams, and interactive assessments using multimedia elements. This would allow institutions to diversify their examination methods, catering to various educational disciplines and learning styles.

Moreover, the Online Exam Panel could incorporate multilingual support to cater to a global audience, making it accessible to students and educators across different regions and languages. Enhancements in mobile compatibility could also be prioritised, ensuring a seamless experience across all devices.

In summary, the Online Exam Panel has great potential for growth, with opportunities to enhance features and integrate advanced technologies, keeping it at the forefront of digital education.

BIBLIOGRAPHY

1) http://www.w3schools.com/

2)https://www.intervue.io/

3) https://projectworlds.in/

4) https://www.researchgate.net/

5) https://stackoverflow.com/

6) https://www.academia.edu/

7) https://www.eklavvya.com/

8) https://online.visual-paradigm.com/

References

1. **MERN Stack**
    - **Official Documentation:** MongoDB, Express, React, Node.js
    - **YouTube Tutorial:** MERN Stack Tutorial for Beginners
2. **Chakra UI**
    - **Official Documentation:** Chakra UI Documentation
    - **YouTube Tutorial:** Chakra UI Crash Course
3. **MUI (Material-UI)**
    - **Official Documentation:** MUI Documentation
    - **YouTube Tutorial:** MUI Material-UI Tutorial
4. **React.js**
    - **Official Documentation:** React Documentation
    - **YouTube Tutorial:** React JS Full Course

5. Node.js
    - Official Documentation: Node.js Documentation
    - YouTube Tutorial: Node.js Crash Course
6. Express.js
    - Official Documentation: Express Documentation
    - YouTube Tutorial: Express.js Tutorial
7. MongoDB
    - Official Documentation: MongoDB Documentation
    - YouTube Tutorial: MongoDB Crash Course
8. Bootstrap
    - Official Documentation: Bootstrap Documentation
    - YouTube Tutorial: Bootstrap 5 Tutorial
9. SweetAlert2
    - Official Documentation: SweetAlert2 Documentation
    - YouTube Tutorial: SweetAlert2 Tutorial
10. React-Redux
    - Official Documentation: Redux Documentation
    - YouTube Tutorial: React Redux Tutorial
11. Moment.js
    - Official Documentation: Moment.js Documentation
    - YouTube Tutorial: Moment.js Tutorial
12. Chakra UI vs MUI
    - Comparison Article: Chakra UI vs MUI
    - YouTube Comparison: Chakra UI vs MUI
13. ChatGPT
    - Official Documentation: OpenAI ChatGPT Documentation
    - YouTube Tutorial: ChatGPT Tutorial for Beginners