AI - POWERED ONLINE ASSESSMENT PROCTORING SYSTEM

By

K. SANTHOSH

(Register No: 111923MC02043)

Of

S.A. ENGINEERING COLLEGE

(Autonomous – Institute Level Research Centre, Affiliated to Anna University, Chennai)

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfilment of the requirements for the award of the degree

Of

MASTER OF COMPUTER APPLICATIONS

ANNA UNIVERSITY

CHENNAI – 600 025

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project report title **"AI - POWERED ONLINE ASSESSMENT PROCTORING SYSTEM"** is the Bonafide work of **Mr. K. SANTHOSH** (Register Number: **111923MC02043**) who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidates.

SIGNATURE

SUPERVISOR

Mr. V. SURESH KUMAR, M.C.A

Assistant Professor (visiting),

Master of Computer Applications,

S.A. Engineering College,

Chennai-600077.

SIGNATURE

HEAD OF THE DEPARTMENT

Dr. V. SUJATHA, M.C.A., Ph.D

Professor & Head,

Master of Computer Applications,

S.A. Engineering College,

Chennai-600077.

Submitted to Project and Viva Examination held on _____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

I am elated to place in record out most sincere appreciation and thanks to our Founder **Thiru. D. SUDHARSHANAM** for providing large facilities for progress.

My sincere and foremost thanks to our honourable Chairman **Thiru. D. DURAISWAMY** for his sincere endeavour in educating me in his premier institutions.

I express my deep sense of gratitude to our beloved Correspondent **Thiru. S. AMARNAATH** and sincere thanks to our Director **Mr. D. SABARINATH** for the facilities provided by them in the college.

I also express my gratitude to our principal **Dr. S. RAMACHANDRAN, M.E., Ph.D.,** who inspired me a lot in completing the project.

I take this opportunity to thank **Dr. V. SUJATHA, M.C.A., Ph.D**., Professor & Head of the Department of Computer Applications, for her cooperation, which has helped me to complete this project.

I am happy to express our sincere thanks to my project internal guide **Mr. V. SURESH KUMAR, M.C.A** Assistant Professor (visiting), Department of Computer Applications, for her constant encouragement and effort taken in directing me throughout the project.

I would like to express my thanks to **SKILL FIRST APPLIED LABS,** who has kindly permitted me to undertake the project in the organization. I am also thankful to all the members of the organization for their support and providing the required information.

I am equally thankful to all faculty members of **MCA** department, family and friends for their endless support throughout the project period.

K. SANTHOSH

CONTENTS

ABSTRACT

The AI-Powered Online Assessment Proctoring System is an intelligent, automated solution designed to maintain the integrity and security of online examinations by detecting and flagging suspicious activities in real-time. Unlike traditional proctoring methods that rely on safe browsers or human invigilation, this system leverages artificial intelligence (AI) for automated violation detection. It incorporates real-time face detection to ensure the presence of the test taker, tab switch monitoring to track any attempt to navigate away from the exam window, noise detection to identify external disturbances, and mobile phone detection to flag unauthorized device usage. Each violation is recorded throughout the test, and upon submission, a proctoring score is generated based on the severity and frequency of violations. This score is then weighted and factored into the final test marks, ensuring fairness in assessment. The system consists of three key user roles: Administrators, Teachers, and Students. Administrators manage users, including teachers and students, with features such as bulk student uploads, while teachers create, assign, and oversee tests, analyse student performance through reports, and modify test settings. Students can participate in tests and access their performance reports. Built on the MERN stack, the system ensures scalability and high performance, efficiently managing multiple exams simultaneously. Initially designed for local deployment, the system has potential for future cloud-based expansion. By integrating AI-driven monitoring, this proctoring system enhances the fairness, security, and transparency of online assessments while minimizing human intervention.

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The AI-Powered Online Assessment Proctoring System is an advanced solution designed to enhance the integrity and security of online examinations. With the rise of online education and remote assessments, traditional examination methods face challenges such as cheating, unauthorized activities, and lack of effective supervision. This system addresses these issues by leveraging artificial intelligence (AI) for real-time proctoring, ensuring a fair, secure, and reliable examination environment. Unlike traditional proctoring methods, such as safe browsers or human invigilation, this system focuses on automated violation detection rather than identity verification. It uses real-time face detection to ensure that a test taker is present in front of the camera throughout the exam. If the system detects the absence of a face, it registers a violation. Additionally, tab switch detection monitors whether the exam tab remains active, counting each instance of tab switching as a violation. The system also includes noise detection, flagging instances of heavy background noise that might indicate external assistance. Furthermore, it employs mobile phone detection, identifying if a phone appears in front of the camera and marking it as a violation.

All detected violations are logged throughout the test, and at the time of test submission, the system compiles a proctoring score based on these violations. This score is then weighted and factored into the final test marks, ensuring that students who engage in suspicious activities receive appropriate penalties. The system automatically generates a detailed report, providing insights into student behaviour during the assessment. The system consists of three primary user roles: Administrator, Teacher, and Student, each with specific functionalities. Administrators manage users, including creating, deleting, and viewing teacher and student records, with an additional bulk upload feature for efficiently adding multiple students. Teachers are responsible for creating, managing, and assigning tests, monitoring student performance through detailed reports, and editing test details when necessary. Students can attend assigned tests while being monitored for violations, view their test reports after completion, and manage their profile information.

1.1 PROBLEM DEFINITION

With the rapid shift towards online education, ensuring the integrity and security of remote assessments has become a major challenge. Traditional proctoring methods, such as human invigilation and safe browsers, are often ineffective, costly, and difficult to scale for large examinations. Cheating, impersonation, tab switching, unauthorized device usage, and external assistance remain significant concerns, compromising the credibility of online tests. Existing solutions that rely on human proctors or biometric verification introduce privacy concerns, high operational costs, and scalability limitations. Additionally, many proctoring systems enforce safe browsers, which restrict user actions but fail to comprehensively prevent cheating, making them inconvenient and easy to bypass. To address these issues, the AI-Powered Online Assessment Proctoring System eliminates the need for safe browsers by providing an automated, AI-driven proctoring solution that detects violations such as face absence, tab switching, excessive noise, and mobile phone usage in real-time. The system logs these violations throughout the test and generates a proctoring score, which is weighted into the final test marks, ensuring fair assessments. By leveraging artificial intelligence and the MERN stack, this system offers a scalable, efficient, and cost-effective alternative to traditional proctoring, minimizing human intervention while maintaining the credibility and fairness of online examinations.

CHAPTER II

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Traditional online proctoring systems rely on human invigilation, safe browsers, and biometric verification to monitor and prevent cheating during remote assessments. Human proctoring involves live monitoring through webcams, which is resource-intensive, costly, and difficult to scale for large exams. Safe browsers restrict test-takers from switching tabs or accessing other applications, but they are inconvenient and can be bypassed using external devices. Some systems incorporate biometric authentication, such as facial recognition, to verify a test-taker's identity, but this raises privacy concerns, requires additional computational resources, and increases costs. Many of these solutions depend on manual review of recorded exam sessions, leading to delays and inefficiencies in result processing. Furthermore, most proctoring solutions are expensive, making them unaffordable for many educational institutions, especially those with limited budgets. The reliance on human intervention and predefined restrictions fails to provide a fully automated and intelligent approach to exam monitoring, making existing systems less effective in ensuring fairness and preventing misconduct.

2.1.1 DISADVANTAGES

- High Cost – Traditional proctoring systems are expensive and not affordable for many educational institutions.
- Scalability Issues – Human proctoring requires significant resources, making it difficult to scale for large exams.
- Manual Review Delays – Many systems require human review of recorded sessions, causing delays in result processing.
- Limited Automation – Heavy reliance on human intervention prevents fully automated monitoring and decision-making.
- Inconvenience for Students – Safe browsers and strict access restrictions create a poor user experience.
- Ineffective Cheating Detection – Traditional methods fail to detect advanced cheating techniques, such as using mobile devices or hidden notes.

2.2 PROPOSED SYSTEM

The AI-Powered Online Assessment Proctoring System is designed to overcome the limitations of traditional proctoring methods by providing an automated, cost-effective, and scalable solution for online exams. Unlike existing systems, it eliminates the need for safe browsers and manual human monitoring, ensuring a seamless experience for students while maintaining exam integrity. The system employs real-time face detection to verify the presence of the test-taker, marking an absence as a violation. It also integrates tab switch detection, which records every instance of a user switching away from the exam window. Additionally, noise detection identifies excessive background noise that could indicate external assistance, while mobile phone detection flags the presence of unauthorized devices in front of the camera. All violations are logged and analysed, contributing to a proctoring score that is factored into the final test results. Built using the MERN stack, the system ensures high performance, scalability, and automation, significantly reducing reliance on human intervention. By providing a fully AI-driven, real-time monitoring solution, this system enhances the fairness, security, and efficiency of online assessments, making it an accessible and practical option for educational institutions.

2.2.1 ADVANTAGES

- Eliminates Safe Browsers – No need for restrictive safe browsers, providing a seamless exam experience.
- Automated Monitoring – Uses AI-driven real-time proctoring, reducing the need for human intervention.
- Cost-Effective – Eliminates the high costs of human proctors and expensive proctoring software.
- Real-Time Face Detection – Ensures the test-taker remains present throughout the exam.
- Tab Switch Detection – Detects and logs instances of students switching away from the exam tab.
- Noise Detection – Flags excessive background noise that could indicate external assistance.
- Mobile Phone Detection – Identifies unauthorized phone usage in front of the camera.
- Fast & Efficient Reporting – Generates detailed proctoring reports immediately after the test.

CHAPTER III
DEVELOPMENT ENVIRONMENT

The Development Environment comprises of hardware requirements and software requirements. The Hardware requirement consists of Processor, Hard Disk, Mouse, RAM and Keyboard. The Software requirement consists of Operating System, Front-end tool, Back-end tool and coding language.

3.1 HARDWARE REQUIREMENTS

Processor and RAM play a crucial role in ensuring the smooth execution of AI-powered proctoring. For the development and efficient functioning of this system, the following hardware requirements have been considered.

Processor	Intel Core i3(8^{th} Gen)/AMD Ryzen 3
RAM	8GB or more
Storage	50GB free space SSD/HDD
Camera	Built-in or external camera
Microphone	Built-in or external microphone

Table: 3.1 Hardware Requirements

3.2 SOFTWARE REQUIREMENTS

Operating System is the major part of software requirements. The Front -end Tool and Back End Tool are used for storing and retrieving the information. The Coding Language is most important in developing the application. For the development of this application, the following software requirements have been considered.

Operating System	Windows 10 and above
Backend	Node.js with Express.js
Frontend	React.js
Database	MongoDB
Browser	Chrome, Firefox or Edge

Table: 3.2 Software Requirements

3.3 SOFTWARE DESCRIPTION

NODE.JS WITH EXPRESS.JS:

Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to build scalable and high-performance server-side applications. It is built on Chrome's V8 JavaScript engine and enables asynchronous, event-driven programming, making it highly efficient for handling multiple requests simultaneously. Node.js is widely used for web development, API services, and real-time applications.

Express.js is a lightweight and flexible web application framework for Node.js that simplifies the process of building web servers and APIs. It provides a robust set of features for routing, middleware integration, and request handling, making it an essential tool for developing backend applications.

ADVANTAGES:

- High Performance: Node.js uses a non-blocking, event-driven architecture, enabling fast and efficient handling of multiple requests.
- Scalability: Suitable for building scalable applications that handle high volumes of concurrent users.
- Lightweight and Fast: Express.js provides a minimalistic framework with powerful routing and middleware support, reducing development complexity.

- Full-Stack JavaScript: Developers can use JavaScript for both frontend and backend, streamlining development.
- Extensive Package Ecosystem: Node.js has a vast npm repository with numerous third-party modules for rapid development.

DISADVATAGES:

- Single-Threaded Limitations: Node.js is single-threaded, which may cause performance issues for CPU-intensive tasks.
- Callback Hell: Complex applications may suffer from deeply nested callbacks, making code harder to manage.
- Security Concerns: Requires careful handling of security vulnerabilities due to its open-source nature.

USES:

- Web Server Development: Express.js is widely used for creating RESTful APIs and backend services.
- Real-Time Applications: Suitable for chat applications, live streaming, and collaborative tools.
- Microservices Architecture: Used for developing microservices-based systems due to its lightweight nature.
- Data Streaming: Node.js efficiently handles real-time data streaming, such as video processing and file uploads.

REACT.JS WITH REDUX:

React.js is an open-source JavaScript library developed by Facebook for building fast, interactive, and scalable user interfaces. It follows a component-based architecture, enabling developers to create reusable UI components that efficiently update and render changes in response to user interactions. React uses a virtual DOM to optimize rendering, making applications highly responsive and performant.

Redux is a state management library that works seamlessly with React to handle application state in a predictable manner. It follows a unidirectional data flow, ensuring that

the state remains consistent across the application. Redux is especially useful in large-scale applications where managing complex state transitions becomes challenging.

ADVANTAGES:

- Component-Based Architecture: React allows developers to build modular and reusable UI components, enhancing maintainability.
- Virtual DOM for Performance: React optimizes UI updates by using a virtual DOM, improving rendering speed.
- Efficient State Management: Redux centralizes the application state, making data flow more predictable and reducing bugs.
- Scalability: Suitable for both small and large applications, ensuring seamless state management as the project grows.
- Strong Community and Ecosystem: React and Redux have extensive community support, rich documentation, and a vast collection of third-party libraries.

DISADVANTAGES:

- Steep Learning Curve: Understanding Redux concepts such as actions, reducers, and the store can be complex for beginners.
- Boilerplate Code: Redux requires writing additional boilerplate code, which may increase development time in small projects.
- Frequent Updates: React and Redux undergo frequent updates, requiring developers to keep up with new features and changes.

USES:

- Single-Page Applications (SPAs): React is ideal for building dynamic, high-performance SPAs with fast navigation.
- State-Intensive Applications: Redux is particularly useful in applications with complex state dependencies, such as dashboards and real-time apps.
- Cross-Platform Development: React can be used with React Native to build mobile applications using the same codebase.
- Enterprise-Scale Applications: Large-scale applications benefit from React's modular approach and Redux's centralized state management.

MONGODB:

MongoDB is a popular open-source NoSQL database designed for high performance, scalability, and flexibility. Unlike traditional relational databases, MongoDB stores data in a JSON-like format called BSON (Binary JSON), allowing for dynamic and schema-less data storage. This makes it an excellent choice for applications requiring flexible and fast data handling, such as web and mobile applications.

MongoDB follows a document-oriented approach, where data is stored in collections of documents instead of tables with fixed schemas. It supports horizontal scaling through sharding, making it ideal for handling large amounts of data across distributed systems. With built-in support for indexing, replication, and aggregation, MongoDB provides a powerful solution for modern application development.

ADVANTAGES:

- Schema Flexibility: Unlike SQL databases, MongoDB allows dynamic and flexible data structures, making it easy to modify data models as requirements change.
- High Performance: MongoDB's indexing and query optimization techniques enable fast read and write operations.
- Scalability: Supports horizontal scaling using sharding, making it suitable for handling large-scale applications.
- Replication & High Availability: MongoDB provides built-in replication through replica sets, ensuring data redundancy and failover support.
- Rich Query Language: Offers powerful querying capabilities, including filtering, aggregation, and full-text search.

DISADVANTAGES:

- No ACID Transactions by Default: Unlike relational databases, MongoDB does not support full ACID (Atomicity, Consistency, Isolation, Durability) compliance for complex multi-document transactions.
- Higher Memory Usage: Since MongoDB stores data in a flexible schema, it can consume more disk space compared to structured relational databases.

- Limited Joins: While MongoDB supports $lookup for joining collections, it is not as powerful as SQL-based relational joins.

USES:

- Web & Mobile Applications: MongoDB is widely used in modern web and mobile apps due to its flexible data model and high performance.
- Real-Time Data Processing: Ideal for real-time analytics, IoT applications, and event-driven architectures.
- Big Data & Cloud Applications: Its ability to scale horizontally makes it suitable for cloud-based distributed applications.
- E-Commerce & Content Management: Frequently used in CMS, product catalogs, and inventory management due to its dynamic schema structure.

CHAPTER IV

SYSTEM DESIGN

4.1 DATA MODEL

Data Model is a set of concepts to describe the structure of the database and certain constraints that the database should obey. The main aim of data model is to support the development of information system by providing the definition and format of data. A data model can be a diagram or flowchart that illustrates the relationships between data. Usually data models are specified in a data modelling language. Although capturing all the possible relationships in a data model can be very time intensive, it's an important step and shouldn't be rushed. Well documented models allow stake holders to identify errors and make changes before any programming code has been written. Data model often use multiple models to view the same data and ensure that all processes, entities, relationships, and data flows have been identified. Data Model can be classified into various evolutions. Some of the evolutions are Hierarchical Model, Network Model, Relational Model, Entity Relationship Model and Object Oriented Model.

The structural part of a data model theory refers to the collection of data structures which make up a data when it is being created. These data structures represent entities and objects in a database model. The manipulation part of a data model refers to the collection of operators which be applied to the data structures.

ROLE OF DATA MODEL

The main aim of data model is to support the development information system by providing the definition and format of data. If this is done consistently across systems then compatibility of data can be achieved. If the same data structures are used to store and access data then different applications can share data. Data model is based on data, data relationship, data semantic and data constraint. A data model provides the details of information to be stored, and is of primary use when the final product is the generation of computer software code for an application or the preparation of a functional specification to aid computer software make or buy decision.

CATEGORIES OF DATA MODEL:

i) Conceptual Data Model
ii) Physical Data Model
iii) Implementation Data Model

- **Conceptual Data Model**: This data model provides the concept that is close to the way many users perceive data.
- **Physical Data Model**: This data model provides the concept that describes the details of how data is stored in the computer.
- **Implementation Data Model:** This data model provides the concept that fall between the above two, balancing user views with some computer storage details.

BENEFITS OF DATA MODEL

- **Clarity and Understanding**: It helps in clearly defining and understanding the structure, relationships, and constraints of data within an organization or system.
- **Efficiency**: By providing a blueprint for data organization, it improves data retrieval, manipulation, and storage efficiency, leading to better system performance.
- **Data Integrity**: Data models enforce rules and constraints, ensuring data accuracy, consistency, and integrity throughout its lifecycle.
- **Standardization**: It promotes standardization of data representation and usage, facilitating seamless integration across different systems and applications.
- **Communication**: It serves as a communication tool between stakeholders, including developers, designers, analysts, and business users, fostering a common understanding of data requirements and usage.
- **Scalability**: Well-designed data models support scalability, allowing systems to handle increasing data volumes and complexity without compromising performance.
- **Decision Support**: It enables better decision-making by providing a structured framework for analysing and interpreting data, leading to actionable insights and informed choices.

4.1.1 MONGODB DATA STRUCTURE

A MongoDB data structure diagram is a visual representation that illustrates the organization and relationships of data within a MongoDB database. MongoDB is a NoSQL database that stores data in JSON-like documents with dynamic schemas, allowing for flexible data structures.

In a MongoDB data structure diagram, collections, documents, and fields are typically represented. Here's how your project's data structure is organized.

COLLECTIONS:

In MongoDB, data is organized into collections. Each collection contains documents that store specific data entities.

- Collection: Admins – Stores administrator details.
- Collection: Logins – Stores authentication data for users.
- Collection: Students – Stores student details and their test progress.
- Collection: Teachers – Stores teacher details and assigned tests.
- Collection: Tests – Stores test details and associated questions.
- Collection: Reports – Stores test reports and AI-proctoring analysis.

DOCUMENTS AND FIELDS:

Each document contains structured data stored in key-value pairs.

Collection: Admins

_id: Unique identifier.

profile: Base64-encoded profile picture.

name: Admin's name.

email: Unique email ID.

phone: Unique phone number.

Collection: Logins

_id: Unique identifier.

user_name: User's name.

email_id: Unique email ID.

password: Hashed password.

role: Defines user role (admin, teacher, or student).

Collection: Students

_id: Unique identifier.

profile: Base64-encoded profile picture.

name: Student's full name.

email: Unique email ID.

phone: Phone number.

department: Department name.

registerNumber: Unique student registration number.

batch: Batch year.

section: Section name.

ongoingTests: Array of tests the student is currently taking.

testId: Reference to Tests collection.

AssignedOn: Date assigned.

start_date, end_date, duration: Test schedule details.

completedTests: Array of completed tests.

testId: Reference to Tests collection.

completedAt: Timestamp.

score: Score obtained.

Collection: Teachers

_id: Unique identifier.

profile: Base64-encoded profile picture.

name: Teacher's full name.

email: Unique email ID.

phone: Unique phone number.

department: Department name.

tests: Array of Test IDs assigned to the teacher.

Collection: Tests

_id: Unique identifier.

testname: Test title.

description: Test description.

teacher_id: Reference to Teachers collection.

start_date, end_date: Test schedule.

duration: Test duration in minutes.

status: Test status (pending, ongoing, completed).

proctor_settings: AI-based settings (e.g., face detection, noise monitoring).

questions: Array of embedded question objects.

questionText: The question.

type: Question type (fill-in-the-blanks, choose-one, etc.).

options: Answer choices.

correctAnswers: Array of correct answers.

marks: Marks allocated.

negativeMarks: Penalty for wrong answers.

image: Base64-encoded question image.

report: Array of Report IDs for test evaluations.

Collection: Reports

_id: Unique identifier.

test_id: Reference to Tests collection.

student_id: Reference to Students collection.

answers: Map of question IDs to selected answers.

score: Test score.

proctoring_report:

noise_score: AI-detected noise level.

face_score: AI-detected face consistency.

mobile_score: AI-detected mobile usage.

tab_score: AI-detected window/tab switching.

duration_taken: Time taken to complete the test.

submitted_at: Submission timestamp.

flags: Array of AI-proctoring detected issues (e.g., multiple faces detected, high noise).

RELATIONSHIPS:

MongoDB supports referencing documents across collections. Key relationships in this project include:

Students.ongoingTests.testId → References Tests._id

Students.completedTests.testId → References Tests._id

Teachers.tests → References Tests._id

Tests.teacher_id → References Teachers._id

Tests.report → References multiple Reports._id

Reports.test_id → References Tests._id

Reports.student_id → References Students._id

4.1.2 ENTITY RELATIONSHIP DIAGRAM:

An Entity-Relationship Diagram (ERD) is a visual representation of a database structure that illustrates how different entities in a system are interconnected. It is a crucial tool in database design as it helps define the relationships between various data components. In the AI-Powered Online Assessment Proctoring System, the ERD depicts the core entities such as Student, Teacher, Test, Report, and Proctoring System, along with their attributes and relationships.

The Student entity represents individuals who take online tests, storing details such as their name, email, phone number, department, batch, and section. Each student can attempt multiple Tests, which are created and managed by the Teacher entity. The Teacher entity includes attributes such as name, email, phone number, and department, and is responsible for preparing and assigning tests. The Test entity contains attributes like test name, description, start and end date, duration, status, and a set of questions. Each test consists of multiple

Questions, where each question may include a text prompt, multiple-choice options, correct answers, marks, and an optional image stored as a base64 string.

Once a student completes a test, a Report is generated, documenting the student's answers, score, and a detailed Proctoring Report that analyzes behaviors such as noise detection, face recognition, mobile usage, and tab-switching activities. The Proctoring System plays a crucial role in monitoring the test environment and flagging any suspicious activities, ensuring the integrity of the online assessment.

The ERD helps in visualizing these relationships and structuring the database efficiently. It ensures that the data is well-organized, preventing redundancy and enhancing system performance. By clearly mapping the connections between students, teachers, tests, and reports, the ERD simplifies database management, supporting the development of a robust and scalable proctoring system.

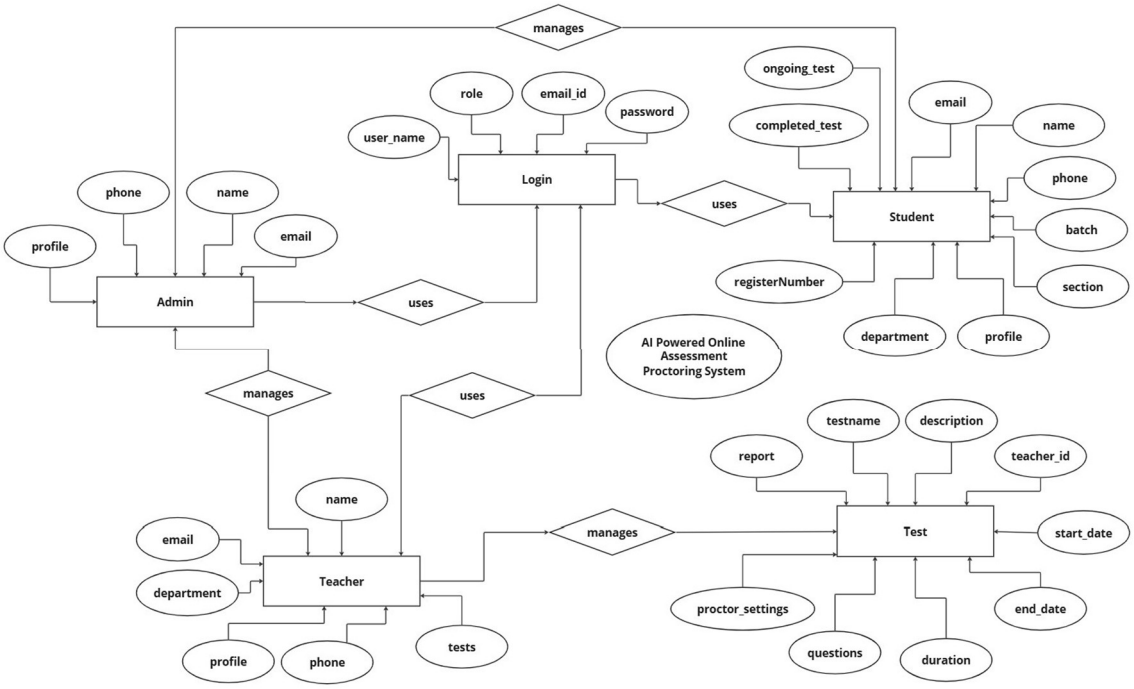

Fig: 4.1 ER Diagram

4.1.3 DATA DICTIONARY:

A data dictionary is a collection of metadata that describes the structure, attributes, and constraints of data elements used in a system. It provides detailed information about each data element, including its name, data type, size, allowed values, relationships, and purpose within the system. The data dictionary serves as a reference for developers, database administrators, and system analysts to ensure consistency and accuracy in data usage across the application.

The data dictionary helps in understanding the data flow within the system and aids in maintaining data integrity. It acts as a centralized repository that defines key entities such as users, tests, reports, and proctoring settings in an AI-powered online assessment proctoring system. It also specifies how data elements are stored, retrieved, and processed.

A data dictionary typically includes elements such as tables, attributes, primary keys, foreign keys, and data formats. It plays a crucial role in database design, system documentation, and application development. By providing standardized definitions, it helps prevent data redundancy, ensures consistency, and facilitates communication between different stakeholders in the project.

Field Name	Data Type	Description
profile	String	profile picture in base64 format string
name	String	name of the admin
email	String	unique email of admin
phone	Number	unique phone number of admin

Fig: 4.2 Data Dictionary Admin

Field Name	Data Type	Description
test_id	Object id	unique id of the test
student_id	Object id	unique id of the student
answers	Object	answers with key value pair
score	Number	score of the test
proctoring_report	Object	Object of proctoring score
duration_taken	String	duration taken to complete test
submitted_at	Date	test submitted date and time
flags	Array	array of flags on proctoring

Fig 4.3 Data Dictionary Report

Field Name	Data Type	Description
email_id	String	unique email of users
password	String	password for login
role	String	role of the user
user_name	String	name of the user

Fig 4.4 Data Dictionary Login

Field Name	Data Type	Description
profile	String	student profile picture in base 64 format
name	String	student name
email	String	unique email of student
phone	Number	unique phone number of student
department	String	department of student
ongoing_test	Array	array of assigned tests
completed_test	Array	array of complted tests
registerNumber	String	Student register number
batch	String	Student batch
section	String	Student section

Fig: 4.5 Data Dictionary Student

Field Name	Data Type	Description
testname	String	name of the test
description	String	description of the test
teacher_id	Object id	unique email id of teacher
start_date	Date	start date of the test
end_date	Date	end date of the test
duration	TimeStamp	duration of the test
proctor_settings	Array	array of proctor settings
report	Array	array of report data

Fig: 4.6 Data Dictionary Test

Field Name	Data Type	Description
profile	String	teacher profile picture in base64 format
name	String	teacher name
email	String	unique email of teacher
phone	Number	unique phone number of teacher
department	String	department of teacher
test	Array	array of test

Fig: 4.7 Data Dictionary Teacher

4.2 PROCESS MODEL

Process models are processes of the same nature that are classified together into a model. Thus, a process model is a description of a process at the type level. The same process model is used repeatedly for the development of many applications and thus, has many instantiations.

Roles:

The various responsibilities and functions assigned to individuals or groups within a process. Assigning roles helps in clarifying who is responsible for specific tasks, making it easier to manage and track progress.

Benefits of Process Models:

- Clarity: Provides a clear visual representation of the process.
- Analysis: Facilitates analysis and identification of bottlenecks or inefficiencies.
- Communication: Helps in communicating the process to stakeholders.
- Improvement: Serves as a basis for process improvement initiatives.

4.2.1 DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a way of representing a flow of a data of a processor a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow. There is no decision rules and no loops. DFD is a graphical representation of the flow of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. It depicts the data flow from one step to another.

Data Flow Diagram Levels:

There are 3 levels in data flow diagram. They are;

- 0-level DFD
- 1-level DFD
- 2-level DFD.

Level 0 DFD

- This is the highest level of abstraction.
- It represents the entire system as a single process (or bubble).
- The external entities (such as users, other systems, or data sources) are shown interacting with the system.
- Arrows represent the flow of data between the system and external entities.

Level 1 DFD

- This level decomposes the single process from the context diagram into subprocesses or functions.
- It provides an overview of the major processes within the system and how they interact.
- External entities may still be present, but the focus is on the internal processes and data flow among them.
- Arrows represent the flow of data between processes.

Level 2 DFD

- This level further decomposes the processes from Level 1 into more detailed sub-processes.
- It provides a detailed view of how data is processed within each of the Level 1 processes.
- Data stores (where data is stored) and data transformations are often represented at this level.
- Arrows represent the detailed flow of data within each process.

Components of Data Flow Diagram:

The Data Flow Diagrams includes four main component elements. They are entity, process, data store and data flow. External Entity also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed.

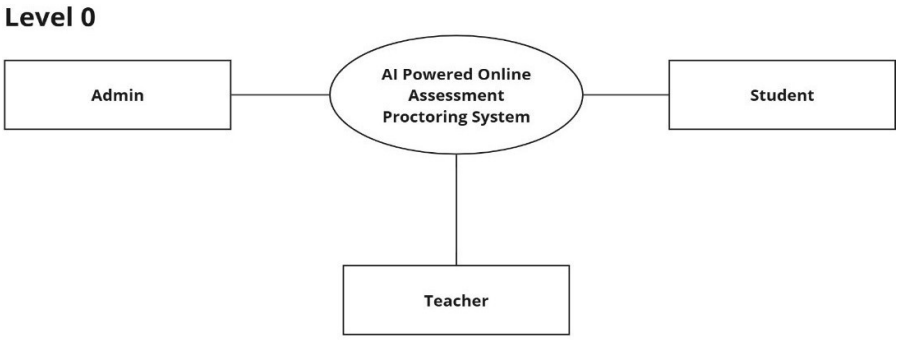

Fig: 4.8 DFD Level 0

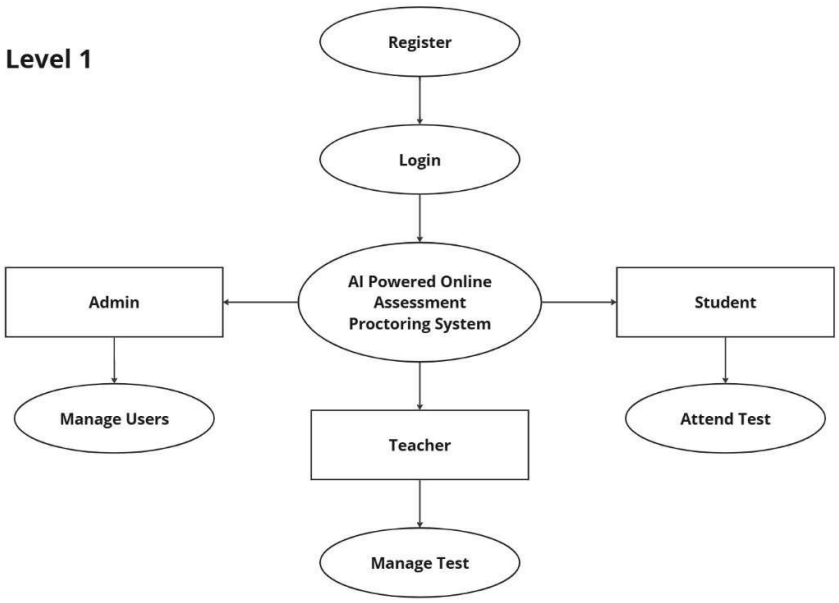

Fig: 4.9 DFD Level 1

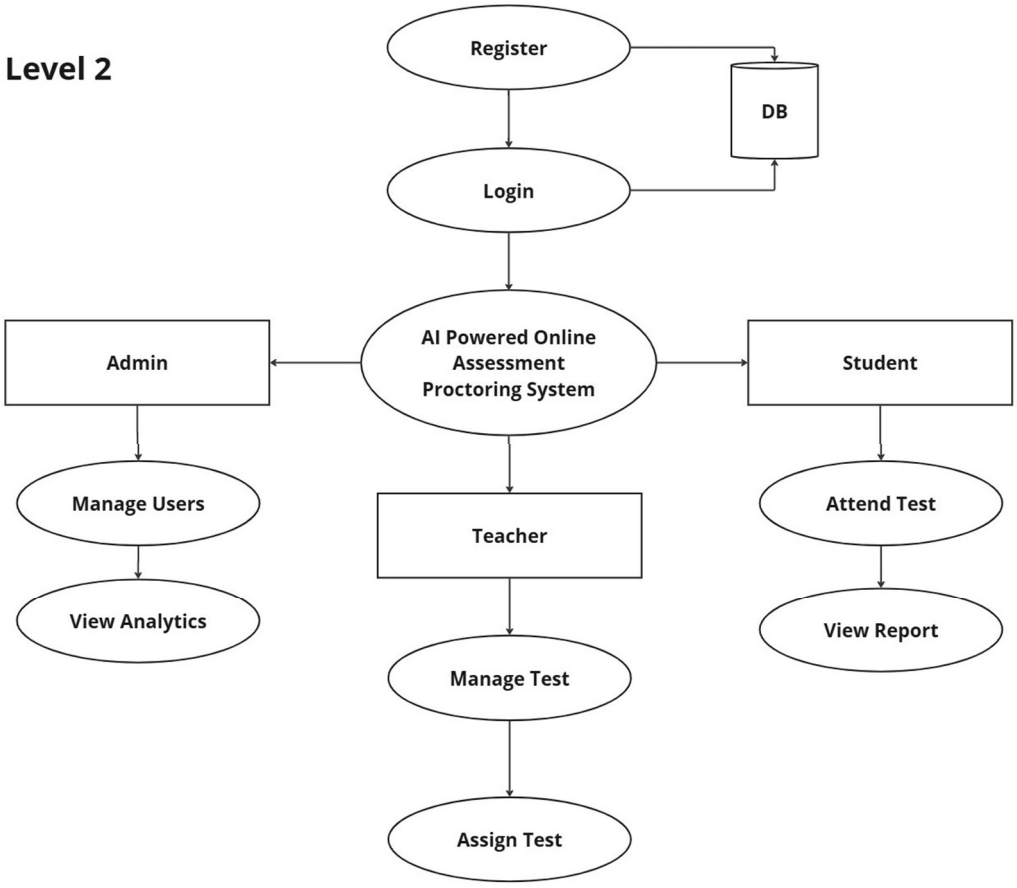

Fig 4.10 DFD Level 2

4.2.2 CONTEXT ANALYSIS DIAGRAM

A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

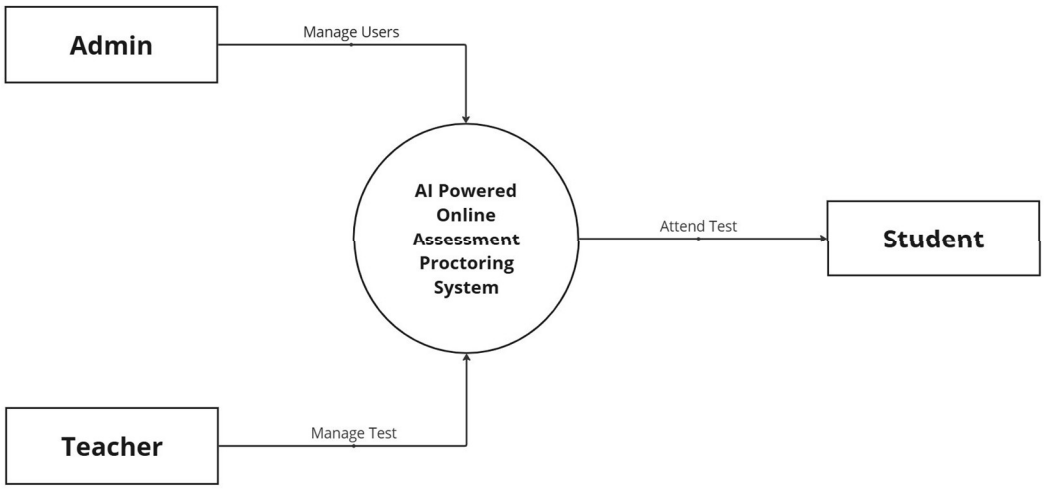

Fig: 4.11 Context analysis diagram

CHAPTER V

SOFTWARE DEVELOPMENT

The Software Development Life Cycle (SDLC) is a structured process used in the software industry to design, develop, and test high-quality software. The goal of SDLC is to ensure that software meets or exceeds customer expectations while staying within time and cost estimates. SDLC provides a detailed methodology for developing, maintaining, enhancing, and replacing software systems, improving both quality and efficiency.

WATERFALL MODEL

The Waterfall Model is a sequential software development approach, where the process flows steadily downwards through several phases, similar to a waterfall. Each phase must be completed before the next begins.

Phases of the Waterfall Model

- Requirements Analysis – Collect and document software requirements.
- Software Design – Create system architecture and design specifications.
- Implementation – Develop the software based on the design.
- Testing – Verify the functionality and fix defects.
- Integration (if applicable) – Combine multiple subsystems.
- Deployment (Installation) – Release the software for end users.
- Maintenance – Address issues, provide updates, and optimize performance.

Key Principles of the Waterfall Model

- The project is divided into sequential phases with minimal overlap.
- Extensive planning, strict timelines, and detailed documentation are emphasized.
- Each phase requires formal approval/sign-off before moving to the next.
- The model discourages revisiting previous phases, making changes difficult once a phase is completed.

5.1 PHASES OF SOFTWARE DEVELOPMENT

The software development life cycle consists of several phases, each playing a crucial role in ensuring the successful development of the AI-powered online assessment proctoring system. These phases include planning, analysis, design, development, testing, implementation, and enhancement. The Waterfall Model is followed for this project, ensuring a structured and sequential approach to development.

PHASE I – PLANNING

The planning phase involves defining the project's objectives, scope, and requirements. This system is designed to enhance the security of online examinations using AI-based proctoring techniques. During this phase, the key focus is on understanding the challenges in online assessments and defining how AI can address issues such as identity verification, cheating prevention, and behavior analysis. The planning phase also includes estimating the time and cost required to complete the project. Since the development timeline is set for one month, efficient integration of AI technologies is prioritized to ensure that the system is both functional and cost-effective. Additionally, the choice of technologies, including React.js, Node.js, MongoDB, and TensorFlow.js, is determined in this phase. Planning plays a vital role as it provides a clear roadmap for development and helps mitigate potential risks before moving forward.

PHASE II – ANALYSIS

The analysis phase focuses on breaking down system requirements and designing the overall system architecture. A detailed study is conducted to understand the specific needs of administrators, teachers, and students, ensuring that the system meets their requirements effectively. The roles of different users are clearly defined, with administrators overseeing the quiz platform, teachers creating and managing quizzes, and students taking exams under AI-powered supervision. During this phase, the database structure is designed to store essential data such as user credentials, quizzes, and AI-generated logs of suspicious activities. Security considerations are also analyzed to ensure that user authentication and data integrity are maintained. The AI proctoring functionality is examined to determine how face detection, head movement tracking, and multiple face detection can be integrated seamlessly into the system.

PHASE III – DESIGN

The design phase involves creating a structured system architecture and user interface layout based on the requirements gathered during analysis. The system is designed with a frontend developed using React.js, allowing for a user-friendly interface, while the backend is built using Node.js and Express.js to handle API requests and database interactions. The database, structured using MongoDB, stores user data, quiz details, and AI-generated logs. The AI component, developed with TensorFlow.js, is designed to perform real-time face detection and behavior analysis to identify potential cheating attempts. The user interface is designed to provide a smooth experience for different user roles, including a dedicated dashboard for administrators, a quiz management panel for teachers, and an AI-proctored exam environment for students. This phase also focuses on defining the interaction between system components, ensuring that data flows seamlessly between the frontend, backend, and AI processing modules. A software specification document is created, outlining the design blueprint, which serves as a reference for developers during implementation.

PHASE IV – DEVELOPMENT

The development phase involves the actual implementation of the system based on the design specifications. The frontend is developed using React.js, ensuring a responsive and interactive interface. Redux Toolkit is integrated for efficient state management, allowing smooth data handling across different components. The backend is built using Node.js and Express.js, with API endpoints created to handle authentication, quiz management, and AI proctoring logs. The database is set up using MongoDB, ensuring efficient data storage and retrieval. AI-powered proctoring functionalities, including face detection, head movement tracking, and multiple face detection, are implemented using TensorFlow.js. The AI models are trained to detect and flag suspicious activities in real-time. Development is carried out in a modular approach, allowing different components to be tested individually before integration. Unit testing and module testing are conducted in parallel to identify and resolve any issues early in the development process.

PHASE V – TESTING

The testing phase is crucial in ensuring that the system functions as expected and meets all security and performance requirements. Unit testing is performed to validate individual components, including login authentication, quiz submission, and AI proctoring functionalities. Integration testing is carried out to verify that all system components interact seamlessly, ensuring that the frontend, backend, and AI processing modules work together without issues. Special attention is given to AI proctoring accuracy testing, where various test cases are executed to confirm that the system correctly detects face absence, multiple faces, and suspicious movements. User acceptance testing is conducted by simulating real exam conditions, allowing teachers and students to interact with the system and provide feedback on usability and performance. Any identified issues are addressed promptly to ensure the system is fully functional before deployment.

PHASE VI – IMPLEMENTATION

The implementation phase involves deploying the system in a real-world environment. Since the initial deployment is planned for a local setup, all necessary dependencies, including MongoDB, Node.js, Express.js, and React.js, are installed and configured. The AI models are integrated into the system, ensuring that real-time proctoring features are optimized for efficient performance. A sample dataset is created, including test quizzes and user accounts, allowing initial testing under practical conditions. Documentation is provided for teachers and administrators, guiding them on how to use the system effectively. The implementation phase ensures that the system is fully operational and ready for regular use.

PHASE VII – ENHANCEMENT & MAINTENANCE

After deployment, continuous monitoring and maintenance are required to address any issues and improve system performance. Bug fixes are implemented as users report errors or unexpected behavior. Performance optimization is carried out to enhance AI response times and improve database efficiency. Security updates are applied to strengthen authentication mechanisms and protect user data. New features are introduced based on user feedback, such as voice detection, screen monitoring, and mobile device compatibility.

5.2 MODULAR DESCRIPTION

The modular description provides a detailed explanation of each module used in the AI-Powered Online Assessment Proctoring System. The key modules in this project are:

- Admin Module
- Teacher Module
- Student Module

5.2.1 Admin Module

The Admin Module is responsible for managing users in the system, including teachers and students. The admin ensures smooth system operations but does not handle quiz creation or monitoring.

- Login – The admin logs in securely using their credentials.
- Manage Teachers – Add, edit, and delete teacher accounts.
- Manage Students – Add, edit, delete student accounts, and handle bulk student uploads.
- View User Details – Check teacher and student profiles, including activity logs.
- View Reports – Access AI-generated proctoring reports and student performance analytics.
- Reset Credentials – Reset passwords or manage access for teachers and students.
- Logout – Securely log out of the system.

5.2.2 Teacher Module

The Teacher Module is responsible for managing quizzes, assigning them to students, and reviewing performance reports. Teachers do not conduct live monitoring but can access AI-generated proctoring reports.

- Login – Teachers authenticate using their credentials to access the system.
- Create Quizzes – Design new quizzes with multiple question formats.
- Edit & Delete Quizzes – Modify or remove quizzes as needed.
- Assign Quizzes to Students – Select students or classes to take a specific test.
- Set Quiz Parameters – Define quiz duration, passing criteria, and allowed attempts.

- View Student Reports – Access AI-generated reports on student performance and flagged activities.
- Export Reports – Download student performance and proctoring logs for review.
- Logout – Securely exit the system.

5.2.3 Student Module

The Student Module allows students to take AI-proctored exams and access their results.

- Login – Students log in securely to access their quizzes.
- View Assigned Quizzes – Check upcoming tests, deadlines, and instructions.
- Take AI-Proctored Exams – Complete exams while the AI monitors their session.
- View Exam Reports – Check their performance, flagged incidents.
- Logout – Securely exit the system after completing exams or viewing results.

CHAPTER VI

TESTING

6.1 SYSTEM TESTING

Testing is the process or group of procedures carried out to evaluate some aspect of a piece of software. Testing plays a vital role in the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. Once program code has been developed, testing begins. The minimum aim of testing process is to identify all defects existing in software product.

Testing establishes the software which has attained a specified degree of quality with respect to selected attributes. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals, that is conducted tests to uncover errors and ensure that defined input will procedure actual results that agree with required results. Testing is related with two processes namely Validation and Verification.

Validation:

Validation is a process of evaluating a software system or component during or at the end of the development cycle in order to determine whether it satisfies specified requirements. It is usually associated with traditional execution based testing.

Verification:

Verification is a process of evaluating a software system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. It is associated with activities such as inspection and reviews of the software deliverable.

TYPES OF SYSTEM TESTING:

- Functional Testing
- Performance Testing
- Stress Testing
- Configuration Testing
- Security Testing

- Recovery Testing

Functional Testing

Functional testing verifies that the system performs as expected, ensuring that inputs produce the correct outputs. It checks whether all required functionalities work correctly under normal and extreme conditions.

Key aspects of functional testing:

- Ensuring correct handling of valid and invalid inputs.
- Verifying that the system meets business requirements.
- Checking whether all user roles can perform their designated tasks.

Application in AI-Powered Online Assessment Proctoring System

In our AI-Powered Online Assessment Proctoring System, functional testing was performed on the following modules:

- **Admin Module:**
  - Creating, reading, updating, and deleting teacher and student accounts.
  - Viewing and downloading test reports in Excel format.
- **Teacher Module:**
  - Creating, updating, and assigning quizzes to students.
  - Viewing and downloading test reports in Excel format.
- **Student Module:**
  - Taking quizzes within the allotted time.
  - Viewing test results and downloading reports in PDF format.
- **Proctoring System:**
  - Face detection to ensure student presence during the test.
  - Behavior analysis to detect suspicious activities such as looking away frequently.
  - Environment monitoring to flag unauthorized persons in the background.
- **Report Generation:**
  - Students can download PDF test reports for self-evaluation.

- Teachers and Admin can download Excel reports to analyze student performance.

Performance Testing

Performance testing evaluates the system's speed, responsiveness, and stability under varying workloads.

Key aspects of performance testing:

- Measuring response time for different actions.
- Testing system scalability under increasing loads.
- Ensuring the system performs optimally under peak conditions.

Application in AI-Powered Online Assessment Proctoring System

Performance testing was conducted to verify the system's responsiveness and efficiency under various loads.

Tested scenarios:

- Multiple students taking quizzes simultaneously without system slowdowns.
- Bulk PDF and Excel report generation without crashes.
- AI-based proctoring running in real-time without performance degradation.

Stress Testing

Stress testing checks how the system behaves under extreme conditions, such as high user load or resource constraints.

Key aspects of stress testing:

- Simulating heavy user traffic.
- Pushing system limits with extensive database queries.
- Identifying potential points of failure.

Application in AI-Powered Online Assessment Proctoring System

Stress testing was performed by simulating high workloads, such as:

- Hundreds of students logging in and taking quizzes at the same time.
- Massive data processing during AI proctoring.
- Generating multiple test reports simultaneously.

The system maintained stability under stress, with minimal lag in performance.

Configuration Testing

Configuration testing evaluates system behavior across different environments, including operating systems, browsers, and hardware setups.

Key aspects of configuration testing:

- Verifying compatibility across multiple platforms.
- Checking performance variations on different devices.
- Ensuring proper rendering in different browsers.

Application in AI-Powered Online Assessment Proctoring System

Configuration testing ensured compatibility with different platforms:

- Operating Systems: Tested on Windows, Linux, and macOS.
- Browsers: Verified functionality in Chrome, Firefox, and Edge.
- Devices: Ensured proper rendering on desktops, laptops, and tablets.

Security Testing

Security testing ensures that the system is protected against threats such as unauthorized access, data breaches, and cyberattacks.

Key aspects of security testing:

- Validating user authentication and role-based access control.
- Preventing SQL injection, cross-site scripting (XSS), and other vulnerabilities.

- Securing sensitive user data with encryption techniques.

Application in AI-Powered Online Assessment Proctoring System

Security testing was performed to protect user data and maintain system integrity:

- Role-Based Access Control: Ensured that only admins, teachers, and students had access to their respective features.
- Data Encryption: Implemented encryption for storing sensitive data such as test results.
- Prevention of Attacks: Verified that the system was resistant to SQL injection, XSS attacks, and unauthorized access.

Recovery Testing

Recovery testing assesses how well the system can recover from failures such as system crashes, hardware malfunctions, or unexpected shutdowns.

Key aspects of recovery testing:

- Testing system behavior after abrupt failures.
- Checking data integrity after system restarts.
- Ensuring smooth recovery of incomplete transactions.

Application in AI-Powered Online Assessment Proctoring System

Recovery testing simulated failures such as:

- Network Disconnection: Ensured the quiz resumed when the connection was restored.
- System Crashes: Verified automatic saving of test progress.
- Server Downtime: Tested system restart mechanisms to recover lost transactions.

6.2 TEST DATA AND OUTPUT

Test data is essential in software testing as it ensures that the system functions correctly under various conditions. It consists of predefined inputs used to evaluate system behavior and verify expected outcomes. Test data plays a critical role in identifying defects, validating system responses, and assessing performance.

Test data can be classified into different categories based on its purpose:

Valid Test Data

Valid test data consists of correct and expected inputs to verify that the system behaves as intended. It ensures that all features work properly under normal operating conditions. In an online assessment system, valid test data includes correctly formatted login credentials, quiz submissions with properly selected answers, and report generation requests that return expected results.

Invalid Test Data

Invalid test data includes incorrect inputs designed to check how the system handles errors and unexpected conditions. This data helps in identifying vulnerabilities and ensuring that the system provides appropriate error messages. Examples of invalid test data include incorrect login credentials, unauthorized access attempts, and attempts to bypass security mechanisms through SQL injection or script-based attacks.

Edge Case Test Data

Edge cases involve testing the system's behavior under extreme conditions, such as maximum input values, simultaneous access by multiple users, or stress testing with a large volume of data. This type of testing ensures that the system remains stable and performs efficiently under peak loads.

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system

configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Some of the phases for unit test planning are:

- Describe Unit Test Approach and Risks.
- Identify Unit features to be tested.
- Add levels of detail to the test plan.

Test No	Test Scenario	Test Input	Expected Output	Actual Output	Result
1	Admin creates a student	Enter valid student details	Student created successfully	Student created successfully	Pass
2	Admin deletes a student	Select a student and delete	Student removed from the system	Student removed from the system	Pass

3	Teacher edits a test	Enter test details and save	Test edited successfully	Test edited successfully	Pass
4	Student submits a test	Answer all questions and submit	Test submitted and report generated	Test submitted and report generated	Pass
5	Proctoring system detects suspicious activity	Student moves away from the camera	Alert generated and stored in report	Alert generated and stored in report	Pass

Table: 6.1 Unit Testing

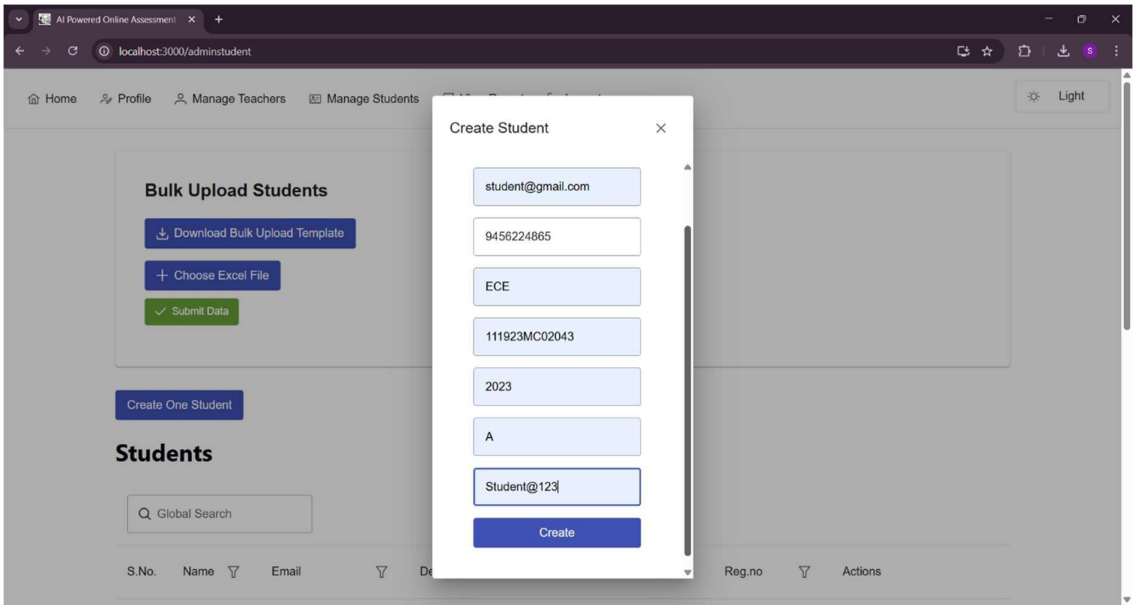

Fig: 6.1 Test 1 Admin Creates Student

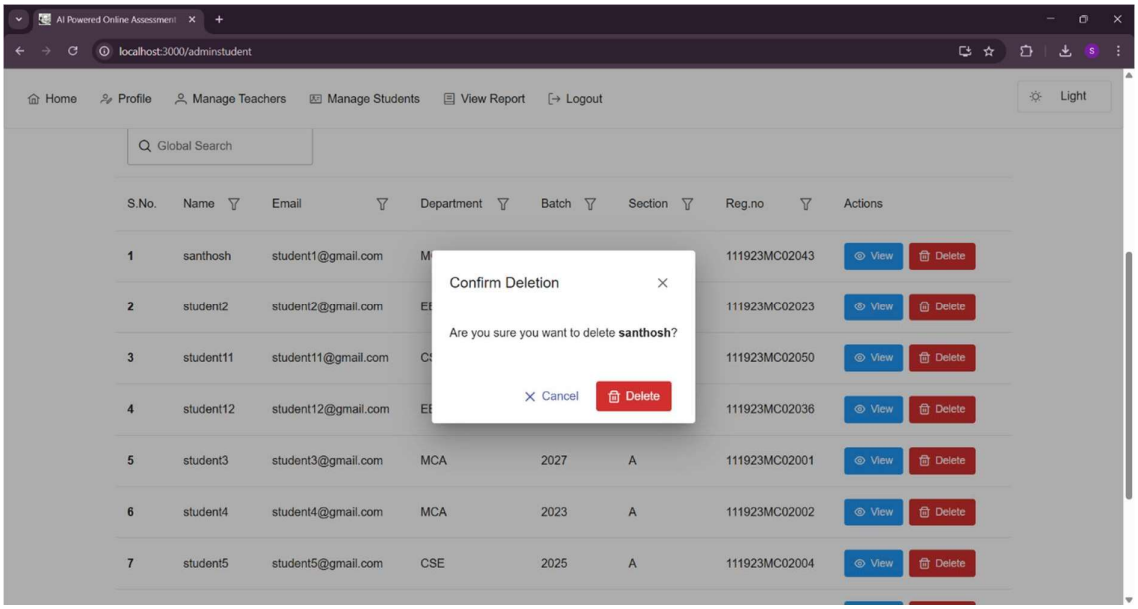

Fig: 6.2 Test 2 Admin Deletes Student

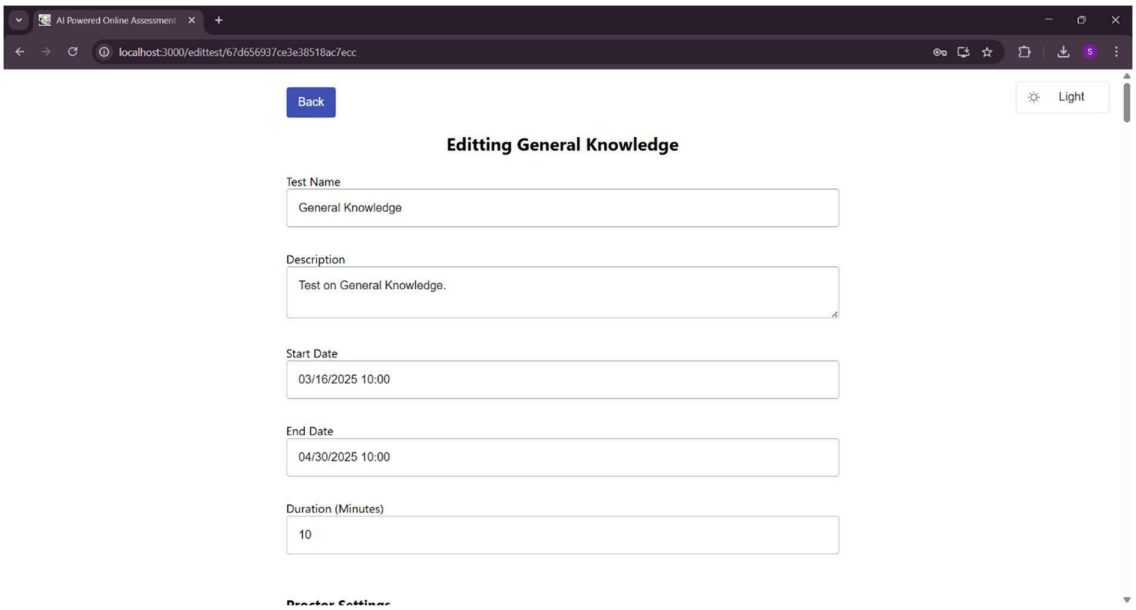

Fig: 6.3 Test 3 Teacher Edits Test

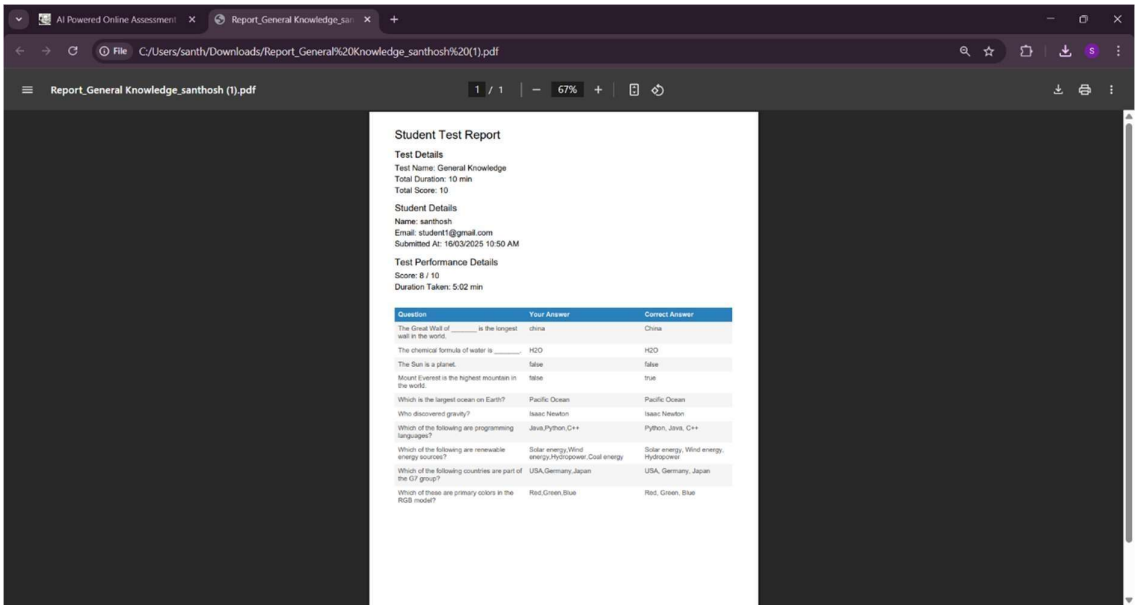

Fig: 6.4 Test 4 Student View Test Report

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in

increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination.

Test No	Test Scenario	Test Input	Expected Output	Actual Output	Result
1	Admin creates a student → Student logs in	Admin enters valid student details	Student can log in successfully	Student can log in successfully	Pass
2	Teacher creates a test → Assigns test to students	Teacher selects students and assigns test	Test assigned successfully and visible to students	Test assigned successfully and visible to students	Pass
3	Student submits test → System auto-generates report	Test submission is completed	Report generated and stored	Report generated and stored	Pass
4	Student attempts test → Proctoring detects suspicious activity	Student engages in prohibited activity	System logs the alert and notifies student	System logs the alert and notifies student	Pass
5	Teacher/Admin views test reports → Downloads report	Select a test report and download	Report downloaded successfully in Excel format	Report downloaded successfully in Excel format	Pass

Table: 6.2 Integration Testing

6.3 TESTING TECHNIQUES / TESTING STRATERGIES

The description of behaviour or functionality for the software under test may come from a formal specification, an Input/Process/Output Diagram (IPO), or a well-defined set of pre and post condition. Another source for information is a requirements specification document that usually describes the functionality of the software under test and its inputs and expected outputs.

6.4 USER ACCEPTANCE TESTING

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

6.5 VALIDATION TESTING

Validation testing takes place after integration testing. Validation testing is a software testing technique that helps ensure that a software application or system meets its requirements and functions correctly. It is a critical part of the software development and quality assurance process. The primary purpose of validation testing is to verify that the software meets the specified criteria and works as intended. Overall, validation testing is a crucial step in the software development lifecycle to ensure that the software product is both functional and meets the user's expectations and requirements. It helps prevent software defects and ensures a high level of quality and user satisfaction.

After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specifications and are accepted.
- Accepted. A validation from specification is uncovered and a deficiency created.

CHAPTER VII

SYSTEM IMPLEMENTATION

7.1 INTRODUCTION

System implementation is a crucial phase in software development, ensuring that the designed system is deployed and operational as intended. It involves translating the theoretical design into a working software application. The implementation process includes requirement analysis, system design, development, testing, deployment, and maintenance.

7.2 IMPLEMENTATION

1. Requirement Analysis

- Gather requirements from key stakeholders, including administrators, teachers, and students.
- Identify essential features such as user registration, test creation, AI-powered proctoring, result evaluation, and report generation.
- Define technical constraints, including scalability, security, and cross-platform compatibility.

2. System Design

- User Interface (UI) Design:
  - Create wireframes and mockups to visualize the layout and navigation.
  - Design intuitive and accessible interfaces for a smooth user experience.
- Database Design:
  - Define the database schema to store user profiles, test details, exam logs, and proctoring data.
  - Choose a database technology like MongoDB, ensuring data consistency and performance.

3. Backend Development

- Develop backend services to handle user authentication, test management, AI-based proctoring, and result processing.
- Implement secure authentication using JWT (JSON Web Tokens).
- Set up the server infrastructure and configure security protocols to ensure data protection.

4. Frontend Development

- Develop a user-friendly interface using React.js for the web-based system.
- Implement UI components that dynamically interact with backend APIs.
- Ensure the frontend is responsive and compatible with different screen sizes.

5. AI-Powered Proctoring Integration

- Implement real-time face detection using TensorFlow.js to monitor test-takers.
- Analyze behavior patterns to detect and flag suspicious activities.
- Ensure the system records logs of anomalies for review by administrators.

6. User Registration and Authentication

- Implement secure login and registration features for administrators, teachers, and students.
- Encrypt user credentials and apply role-based access control.

7. Test Creation and Management

- Allow teachers to create tests with multiple question types, including text, code-based, and image-based questions.
- Enable setting up exam time limits and configuring AI-proctoring rules.

8. Exam Monitoring and Result Processing

- Enable real-time exam monitoring with AI-assisted flagging of suspicious behavior.
- Automate result evaluation and provide reports for teachers and administrators.

9. Testing

- Perform unit testing to validate individual components.
- Conduct integration testing to ensure seamless communication between frontend, backend, and AI-proctoring modules.
- Execute user acceptance testing (UAT) to confirm the system meets stakeholder expectations.

10. Deployment

- Deploy the MERN stack application on a local environment.
- Configure security settings, ensuring the system runs efficiently.

11. Maintenance

- Provide ongoing support, addressing bug fixes, optimizations, and updates.
- Collect feedback from users and improve the system based on insights.

CHAPTER VIII

PERFORMANCE AND LIMITATIONS

8.1 MERITS OF THE SYSTEM

- The AI-powered proctoring system enables students to take assessments from any location, reducing the need for physical examination centers.
- Teachers can create and manage online tests efficiently, streamlining the assessment process.
- The system provides real-time monitoring using AI-based face detection and behavior analysis, ensuring exam integrity.
- Automated result evaluation reduces the need for manual grading, saving time and effort.
- The system enhances security by detecting and flagging suspicious activities, reducing the chances of cheating.

8.2 LIMITATIONS OF THE SYSTEM

- The system requires a stable internet connection, which may affect users in areas with poor network coverage.
- AI-based proctoring may generate false positives, leading to unnecessary flags for students exhibiting normal behavior.
- The current implementation does not support real-time student-teacher interaction during the exam.
- The system primarily relies on facial recognition, which may not be effective in cases where lighting conditions or camera quality is poor.

8.3 FUTURE ENHANCEMENTS

- Enhance the AI model to improve accuracy in detecting suspicious behavior, reducing false positives.
- Introduce voice detection to monitor verbal communication and prevent unauthorized assistance.
- Optimize the system to function effectively even with low-bandwidth internet connections.

CHAPTER IX

APPENDICES

9.1 SAMPLE SCREENS AND REPORTS

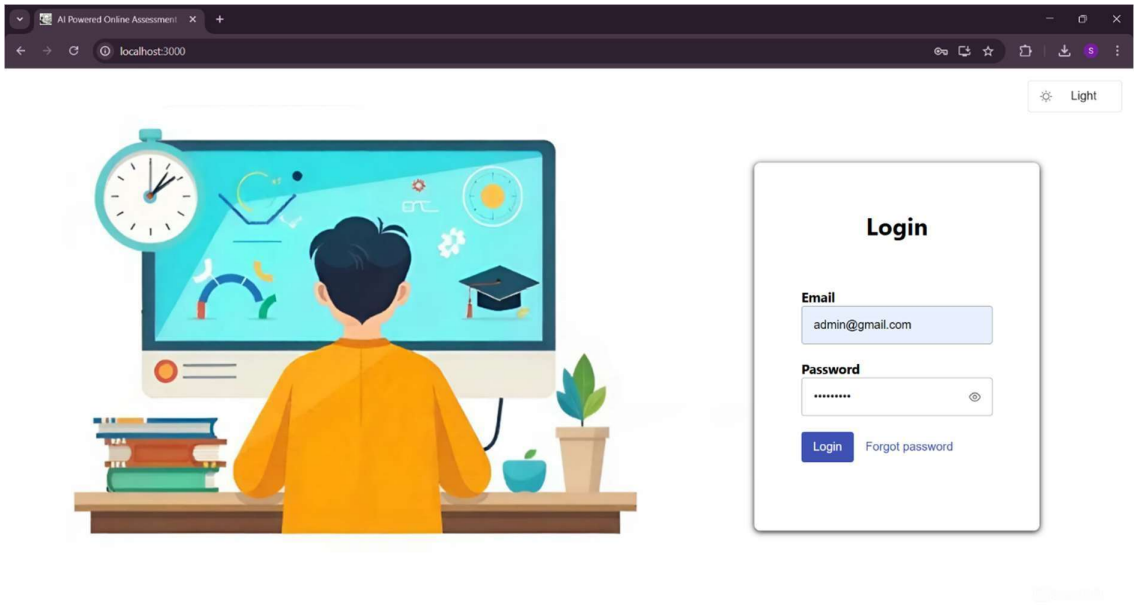

Fig: 9.1.1 All User Login Page

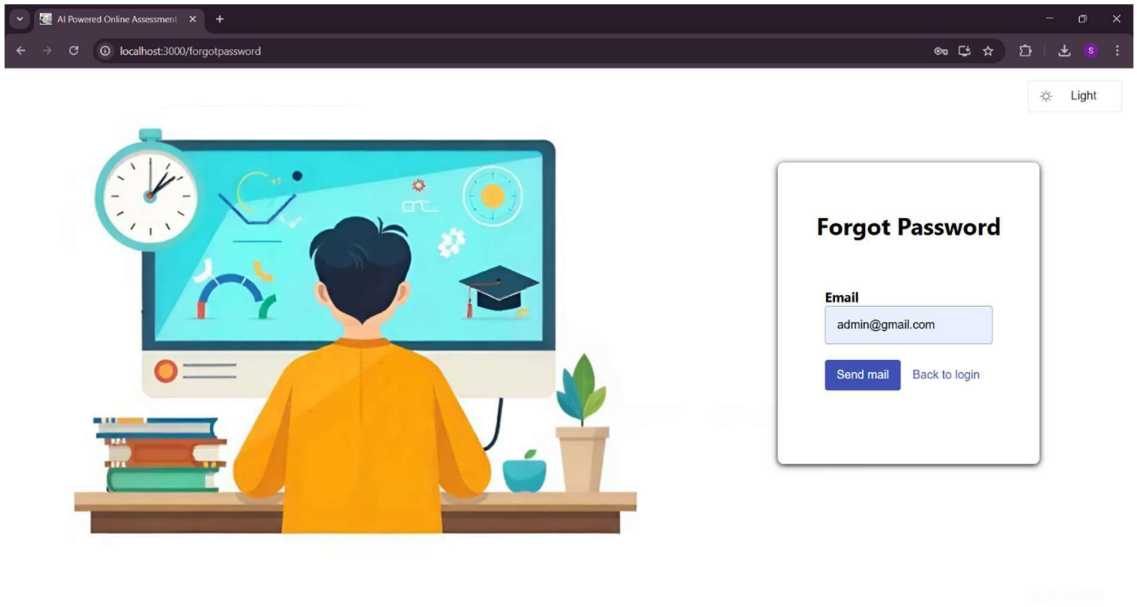

Fig: 9.1.2 All User Forgot Password page

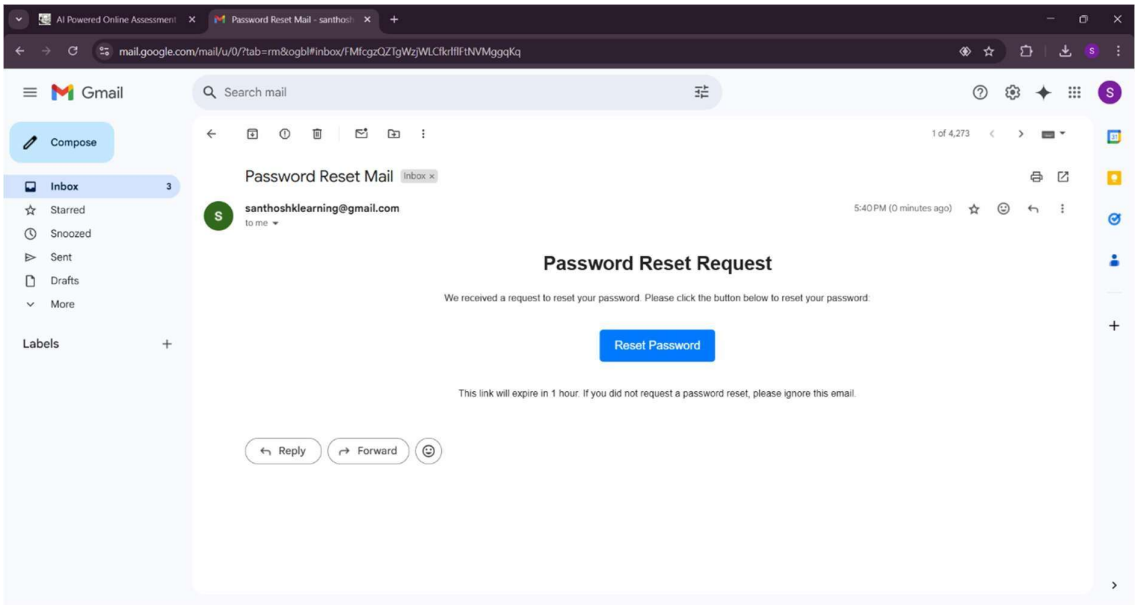

Fig: 9.1.3 Password Reset Mail

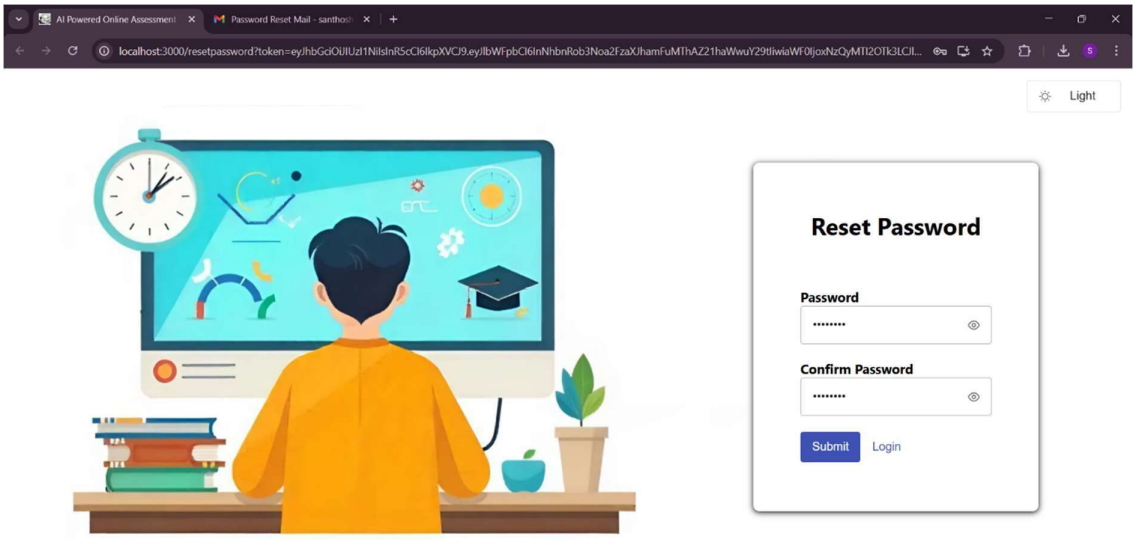

Fig: 9.1.4 All User Reset Password Page

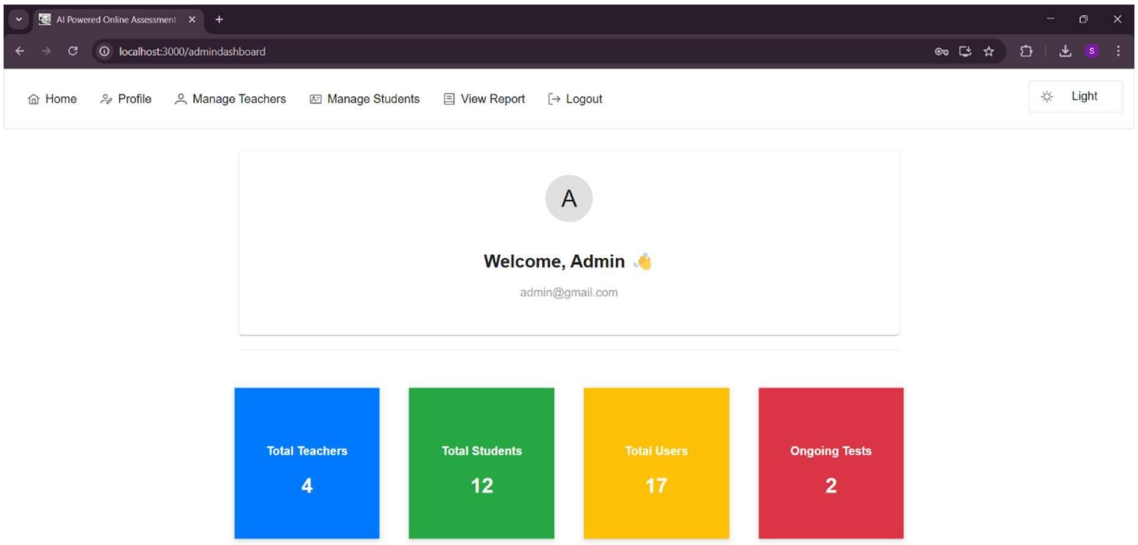

Fig: 9.1.5 Admin Dashboard

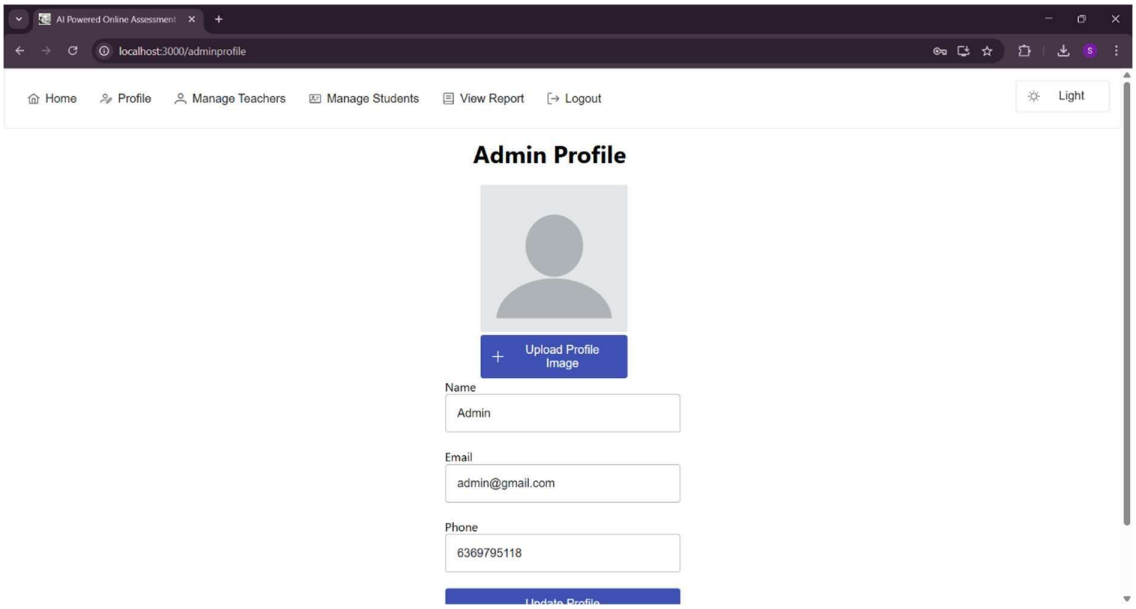

Fig: 9.1.6 Admin Profile

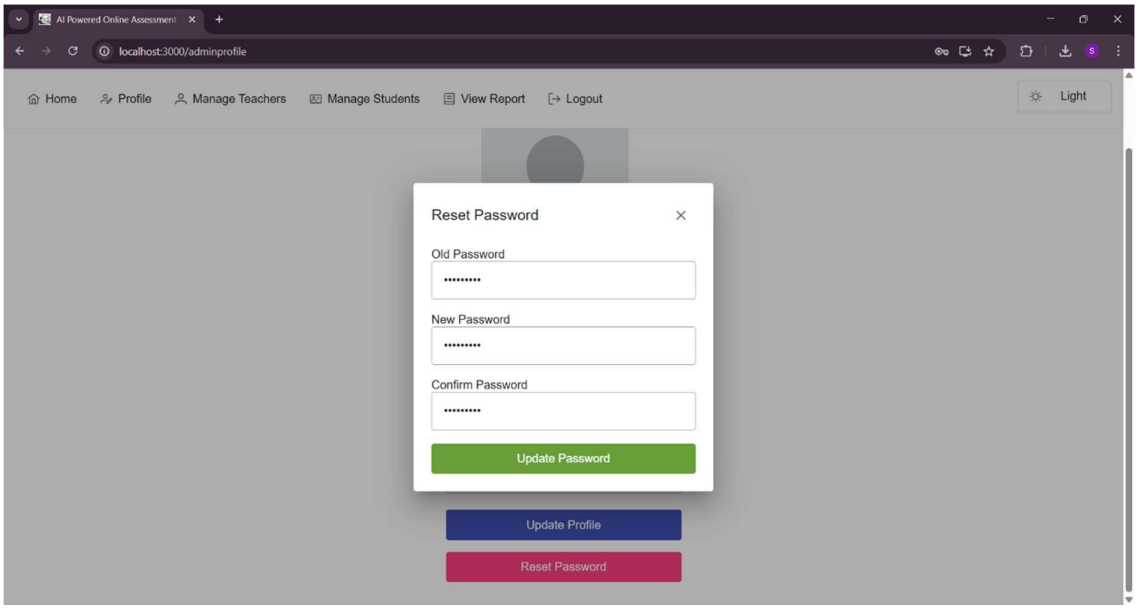

Fig: 9.1.7 Admin Update Password from Profile Page

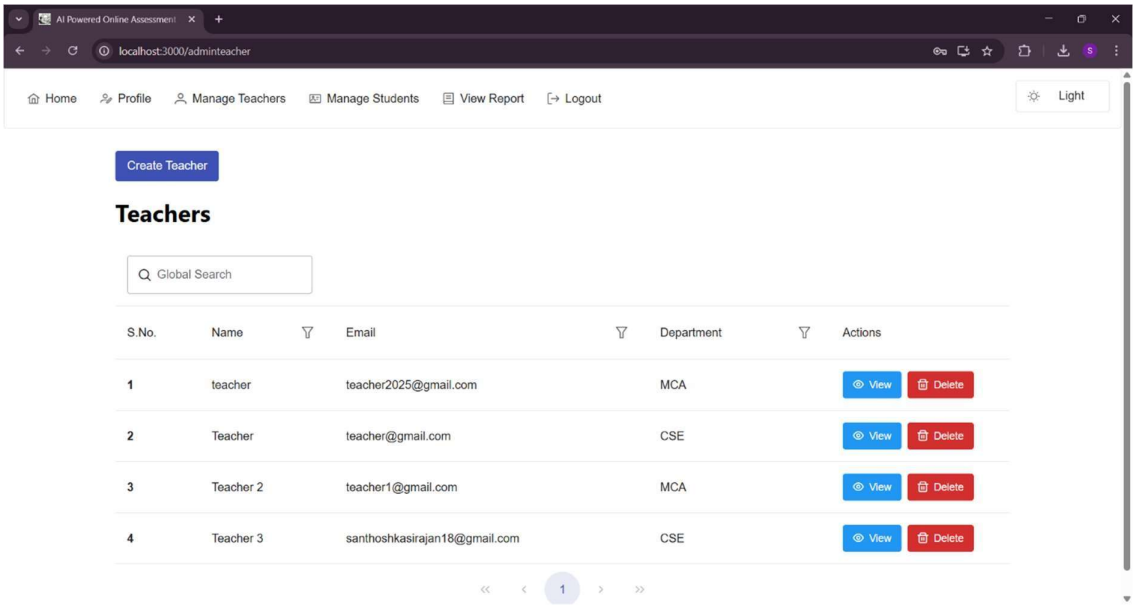

Fig: 9.1.8 Admin Manage Teacher

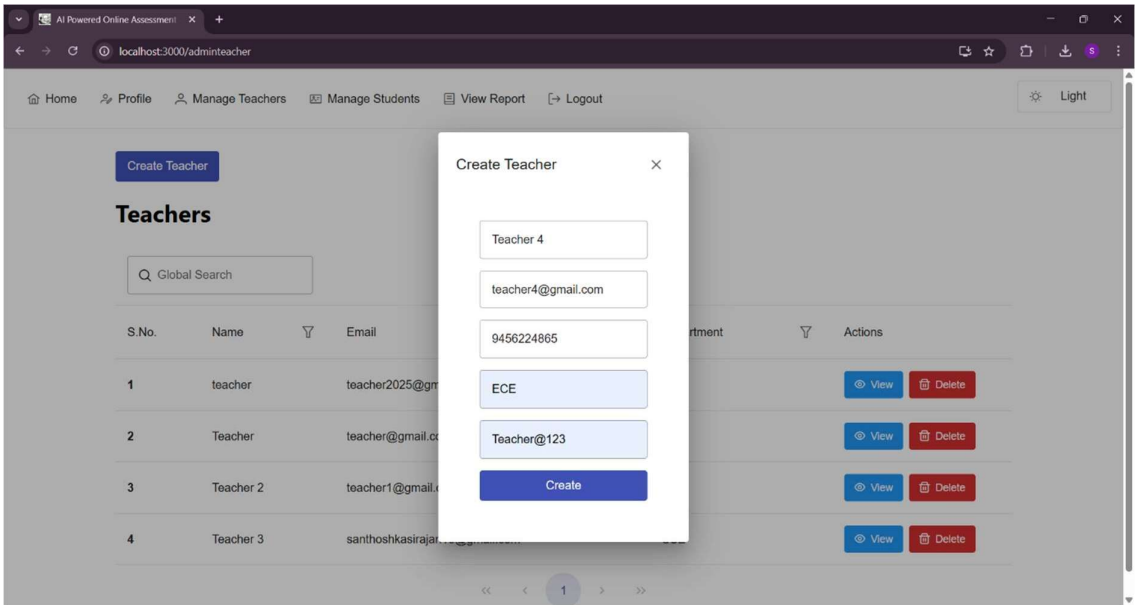

Fig: 9.1.9 Admin Add Teacher

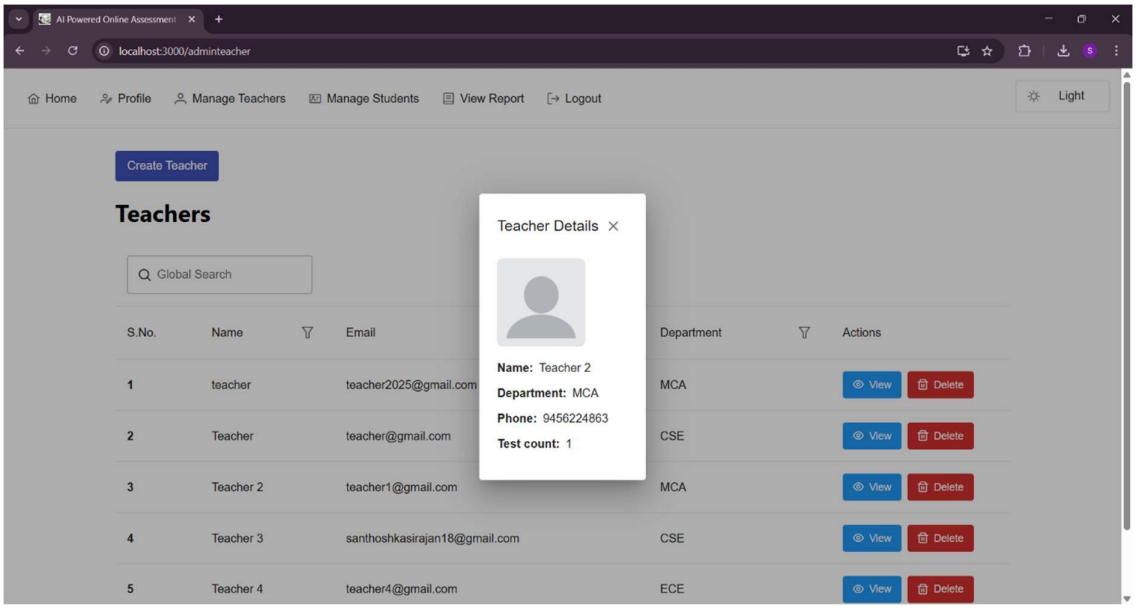

Fig: 9.1.10 Admin View Teacher

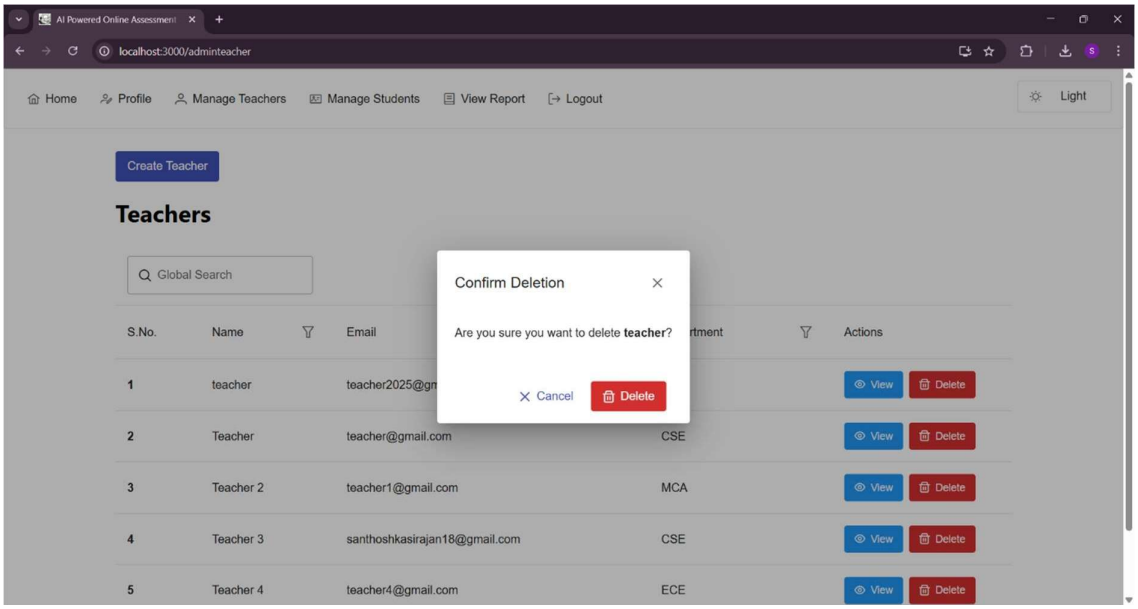

Fig: 9.1.11 Admin Delete Teacher

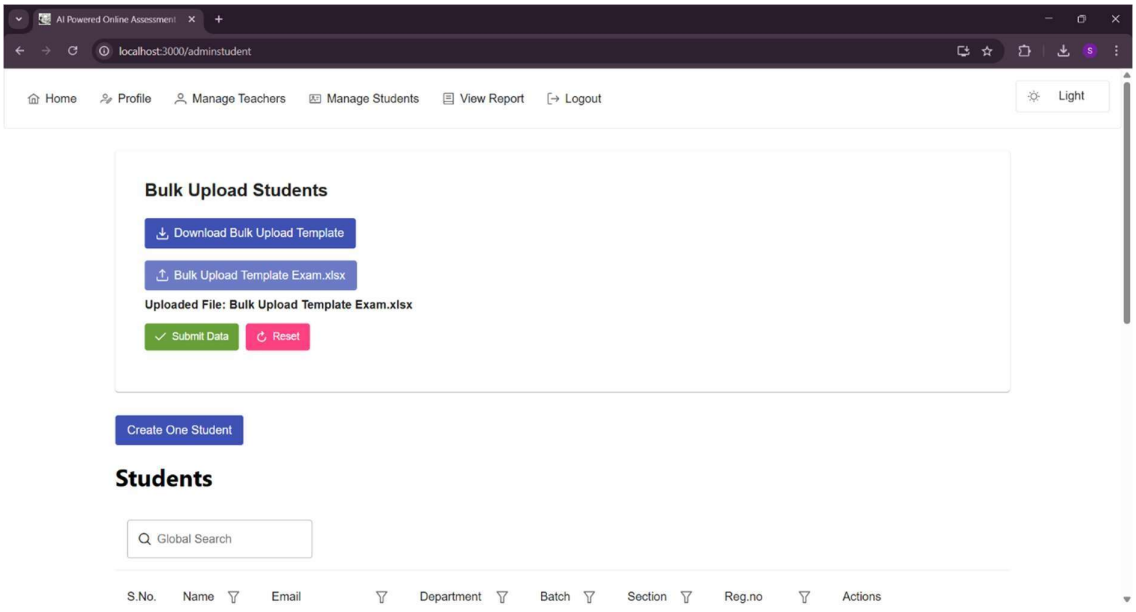

Fig: 9.1.12 Admin Manage Students

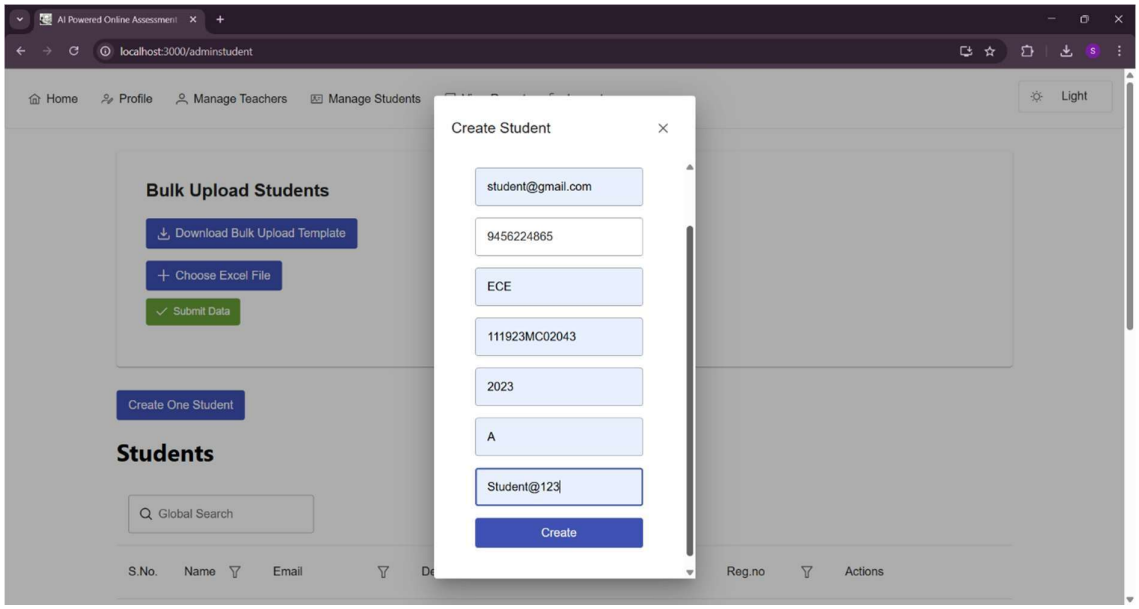

Fig: 9.1.13 Admin Create One Student

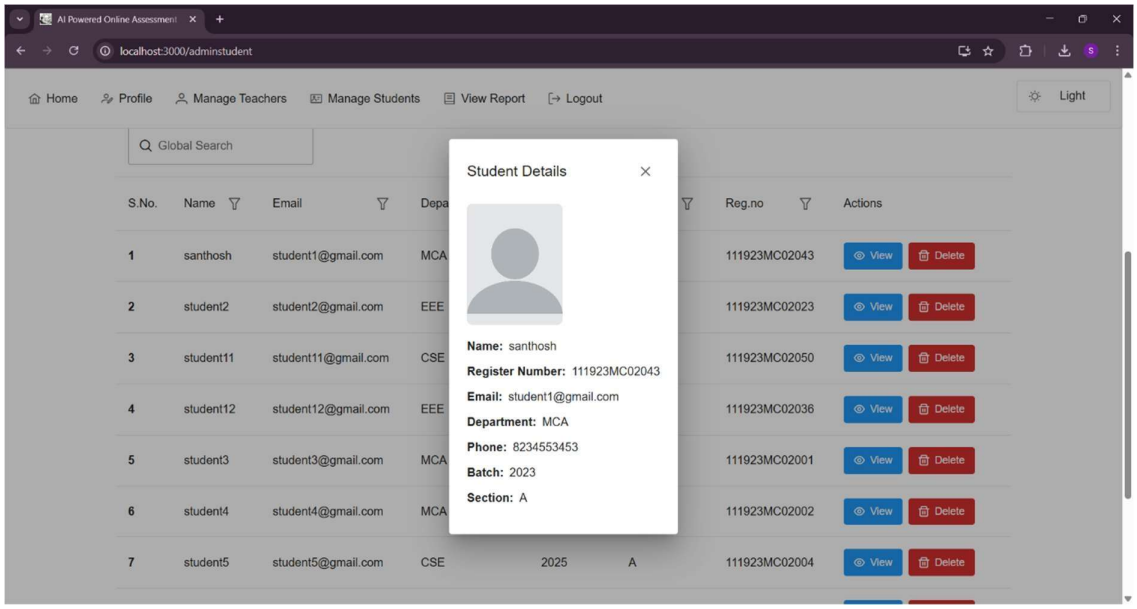

Fig: 9.1.14 Admin View Student

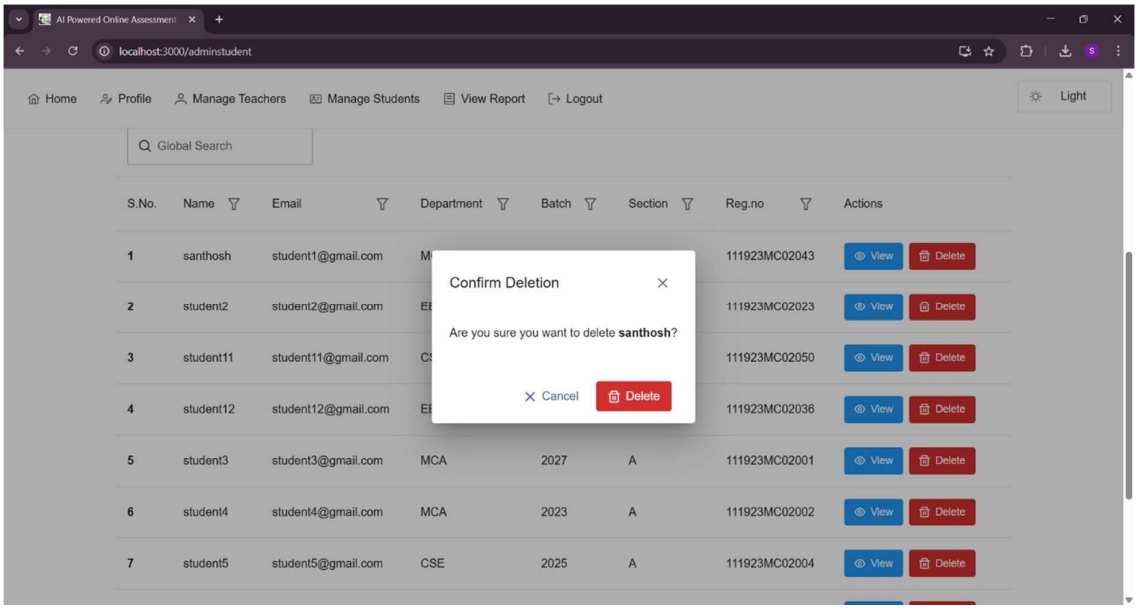

Fig: 9.1.15 Admin Delete Student

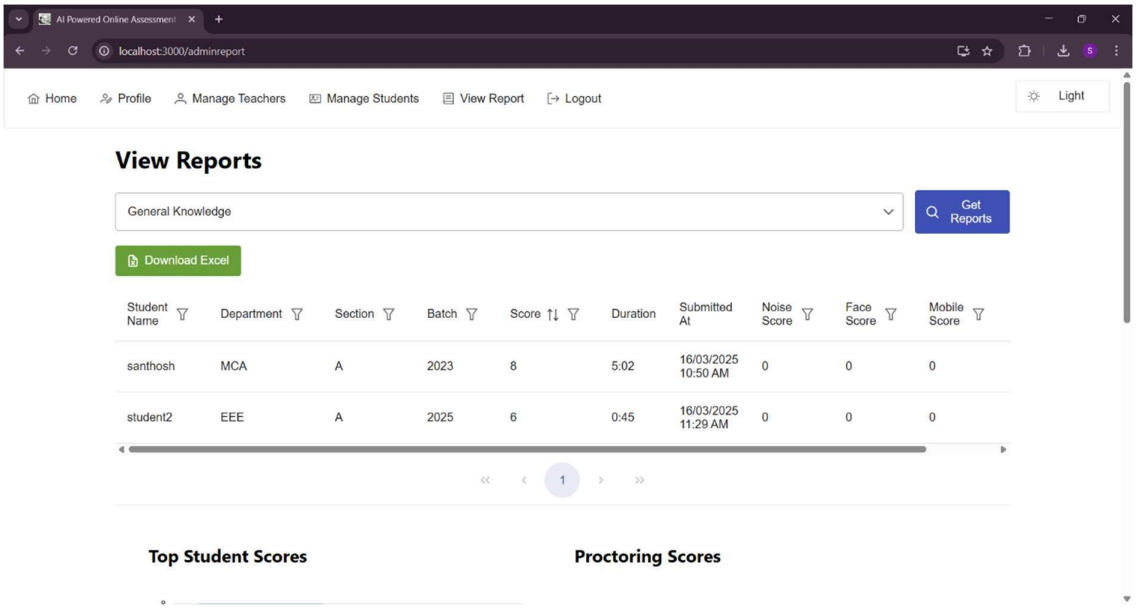

Fig: 9.1.16 Admin View Reports

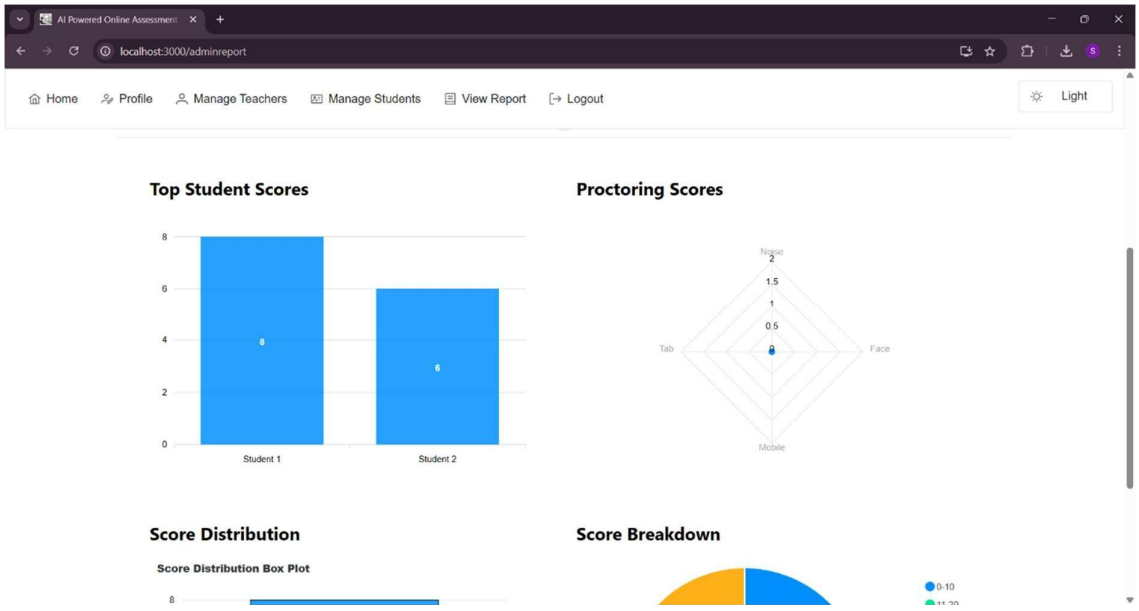

Fig: 9.1.17 Admin View Reports (1)

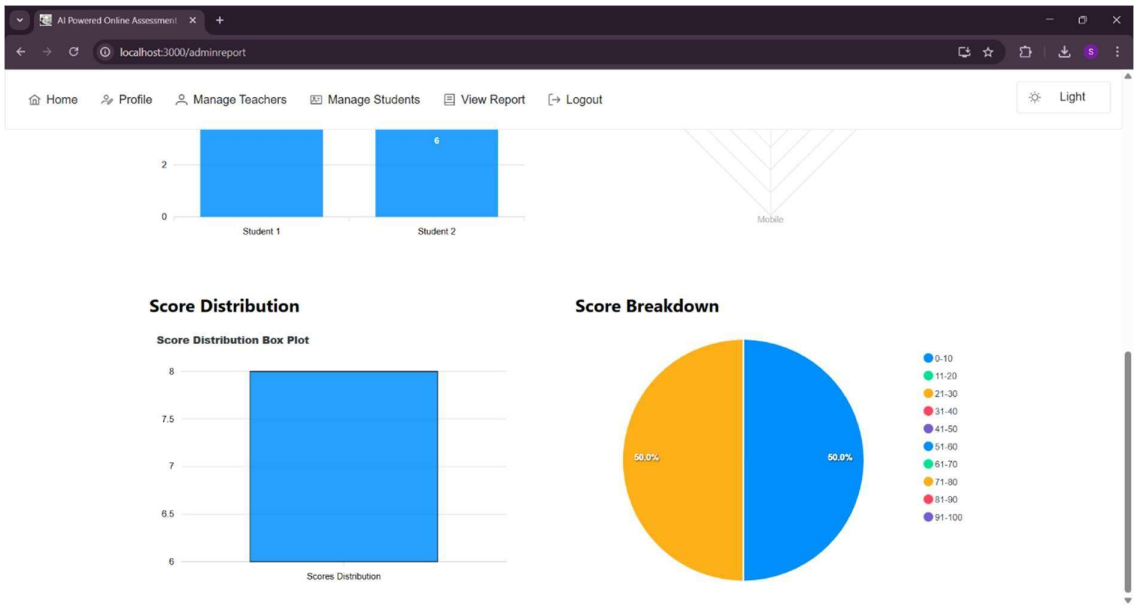

Fig: 9.1.18 Admin View Reports (2)

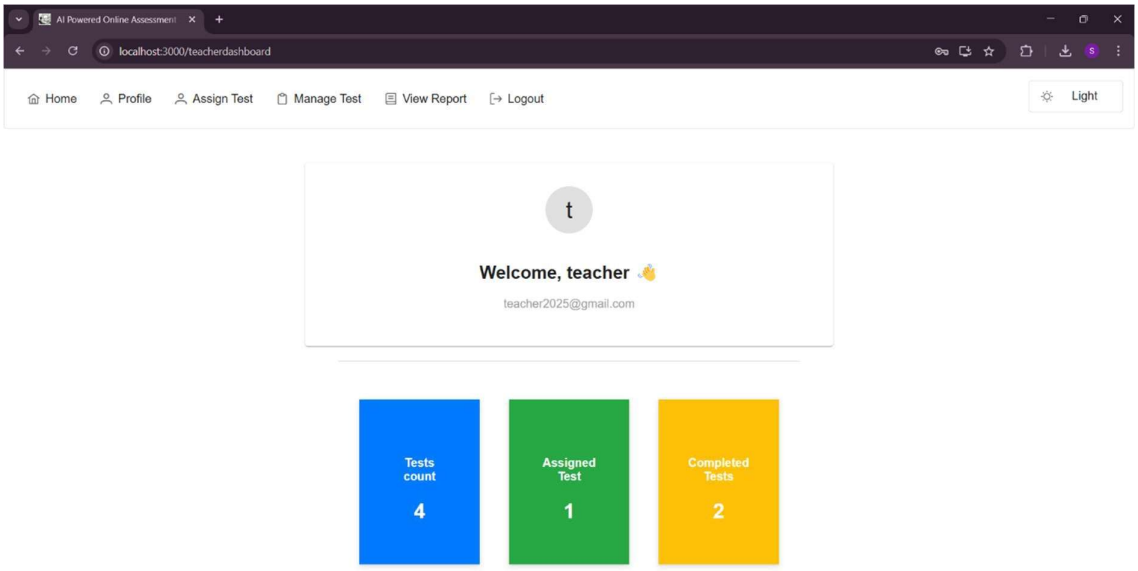

Fig: 9.1.19 Teacher Dashboard

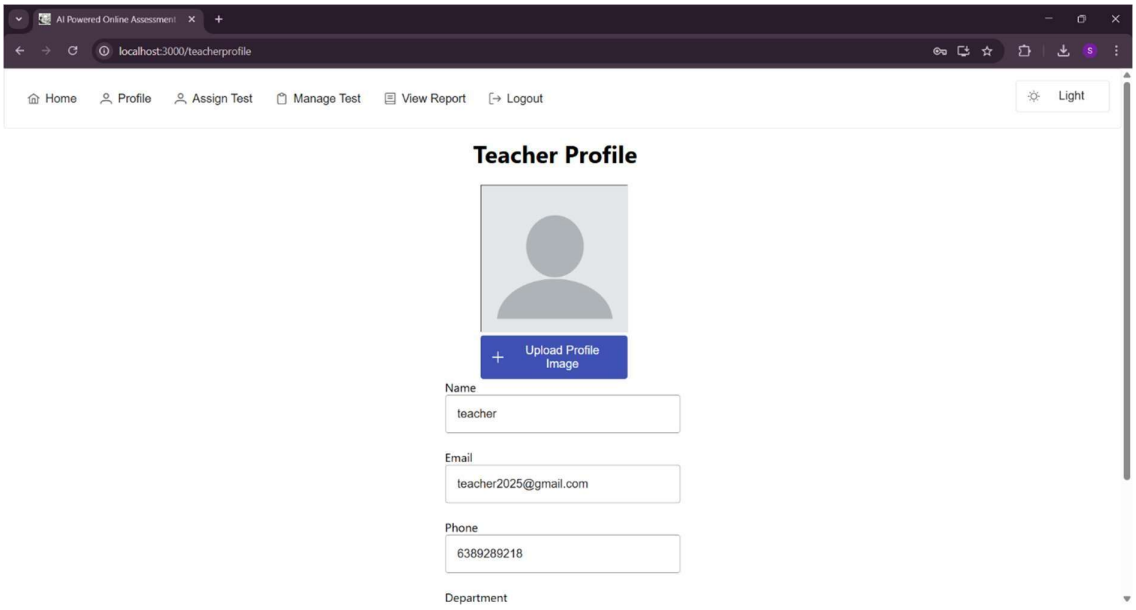

Fig: 9.1.20 Teacher Profile

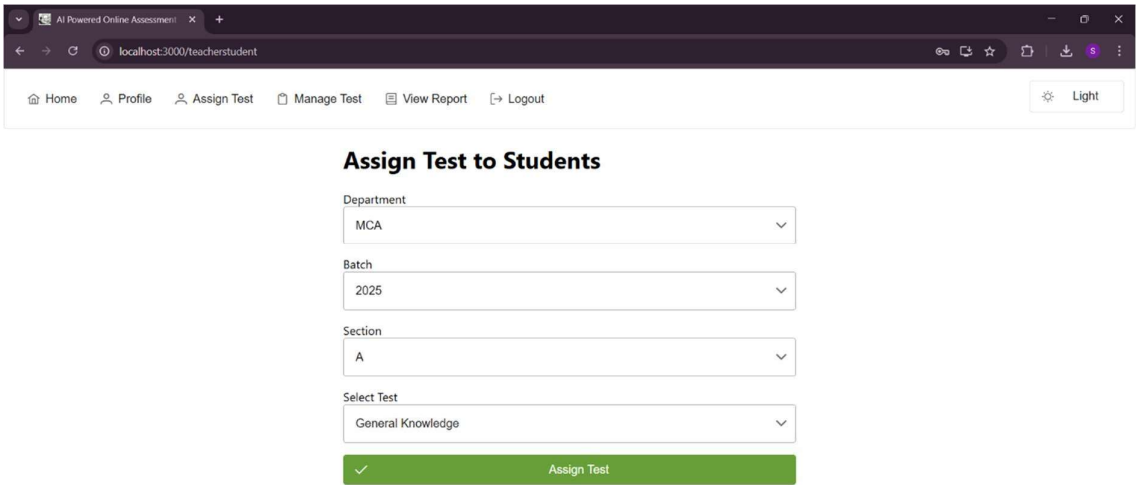

Fig: 9.1.21 Teacher Assign Test

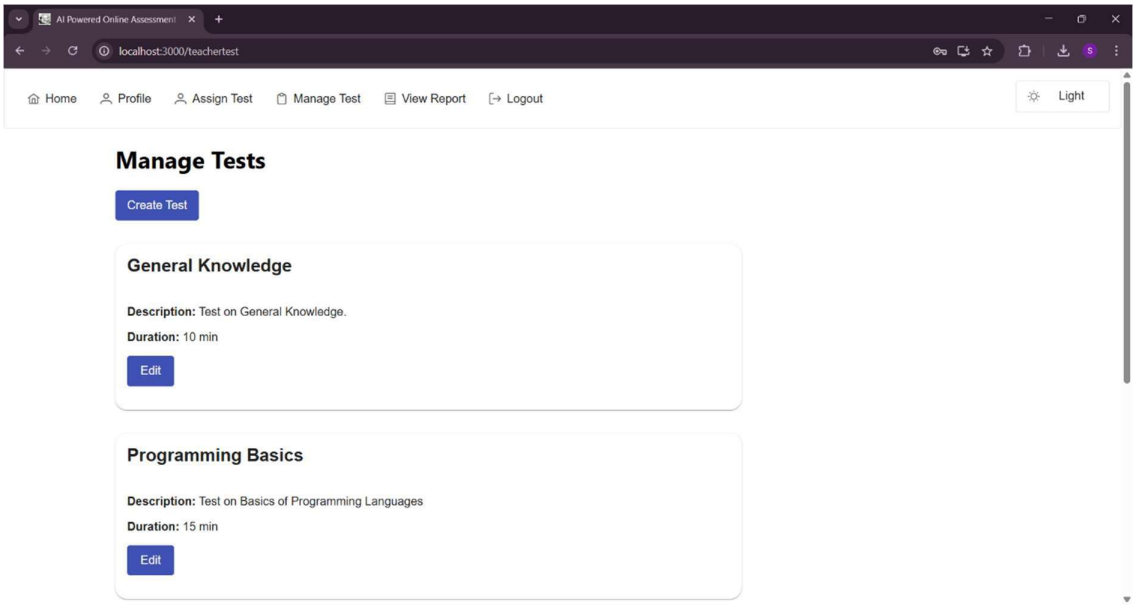

Fig: 9.1.22 Teacher Manage Test

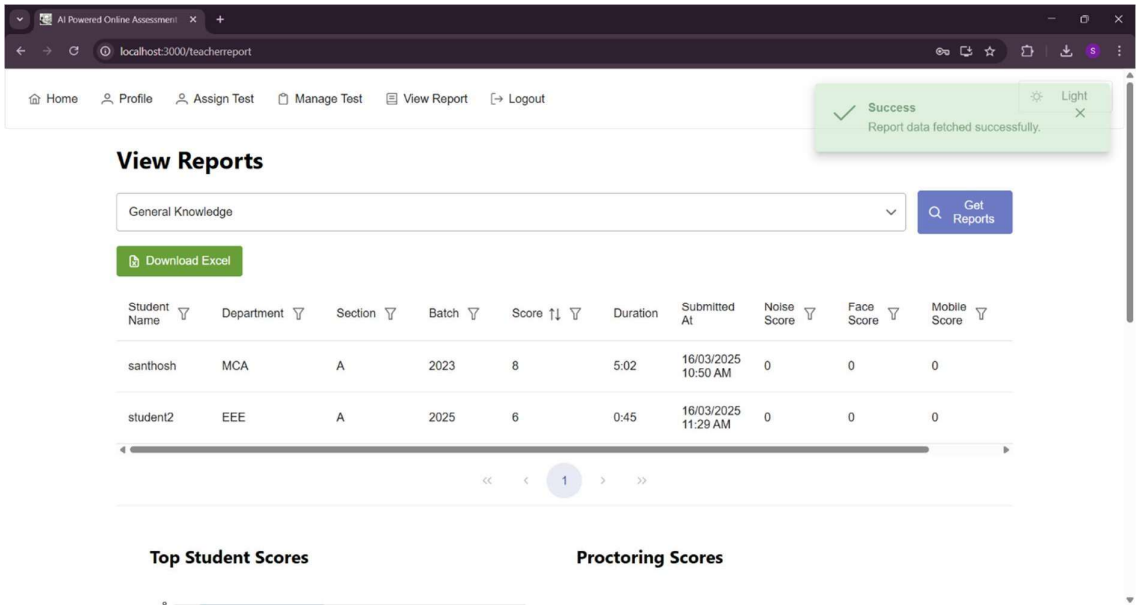

Fig: 9.1.23 Teacher View Report

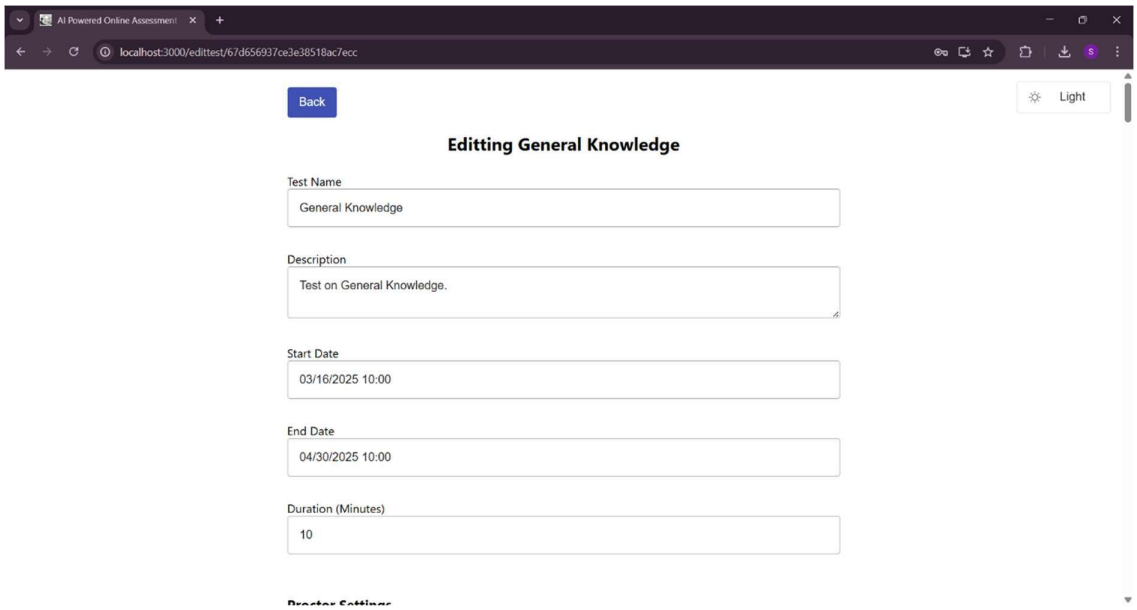

Fig: 9.1.24 Teacher Edit Test

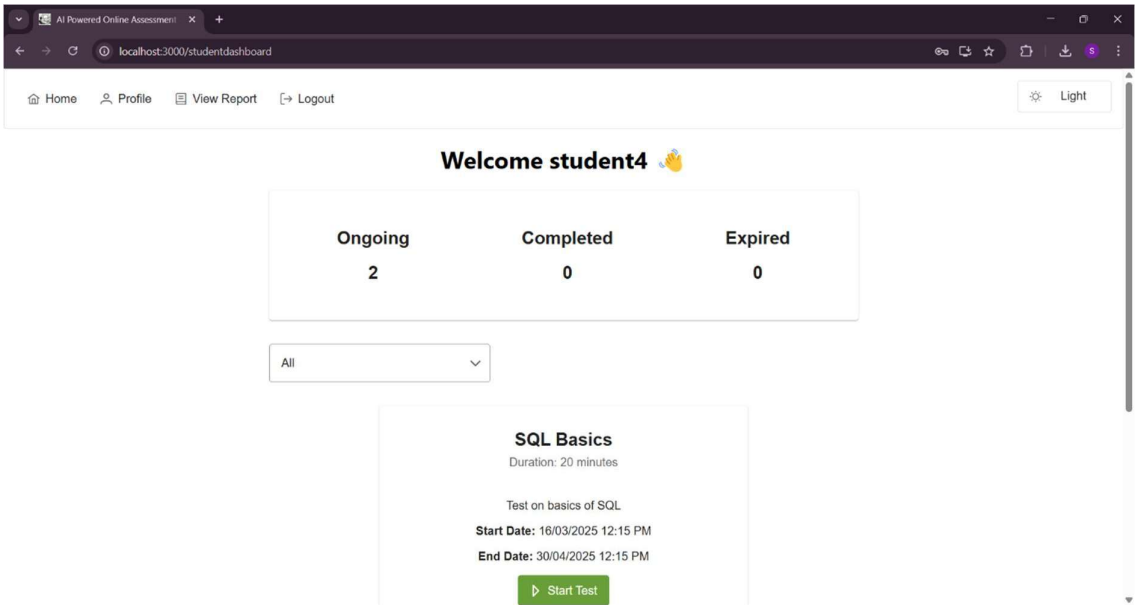

Fig: 9.1.25 Student Dashboard

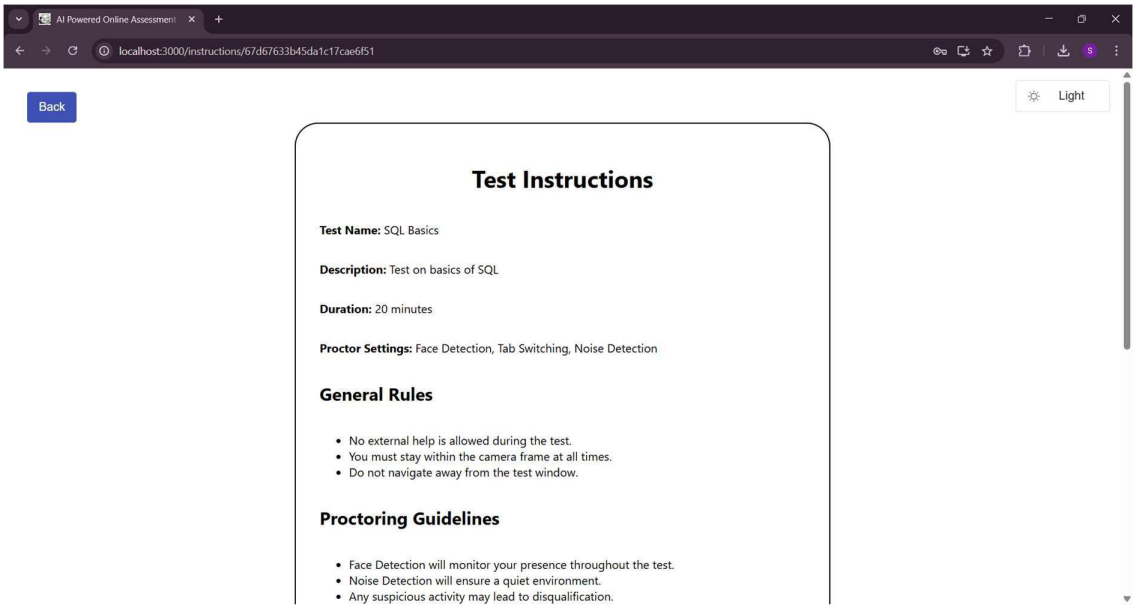

Fig: 9.1.26 Student Start Test

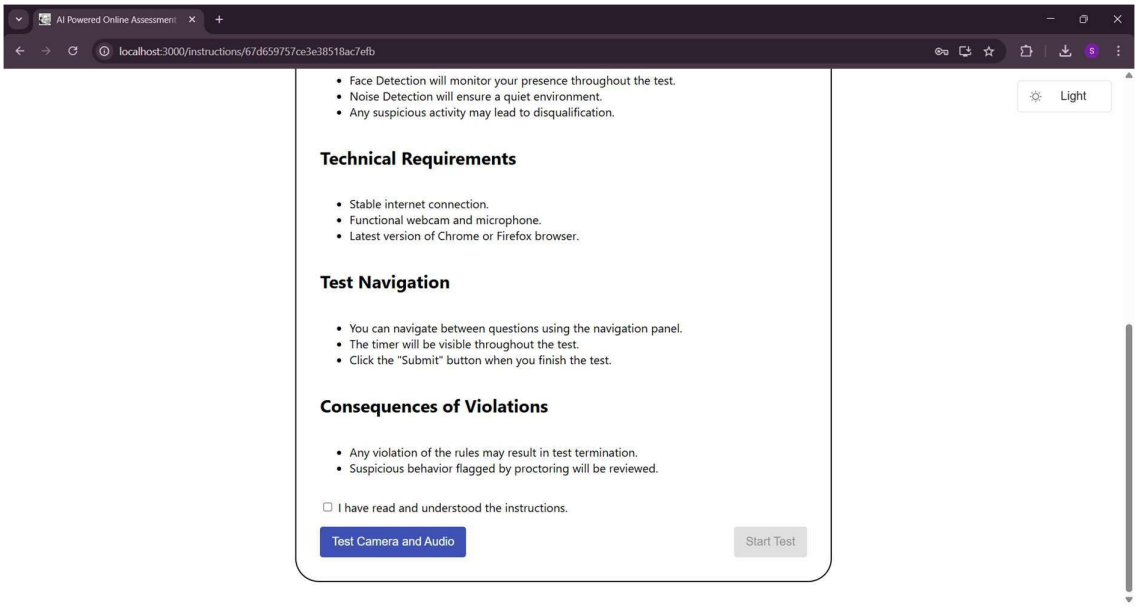

Fig: 9.1.26 Student Start Test

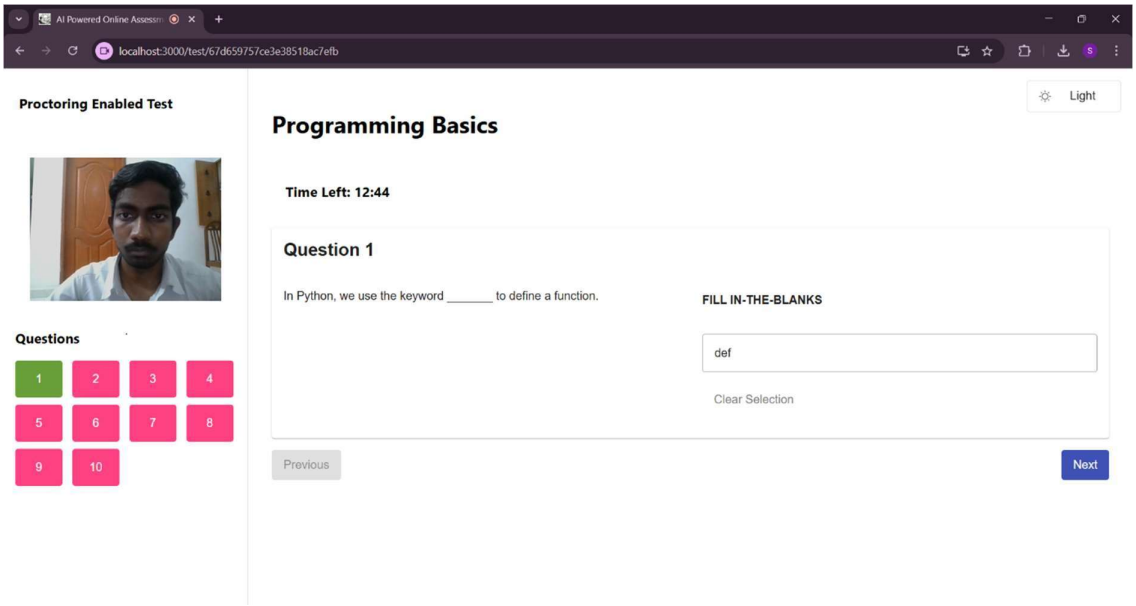

Fig: 9.1.27 Fill in the blanks question type

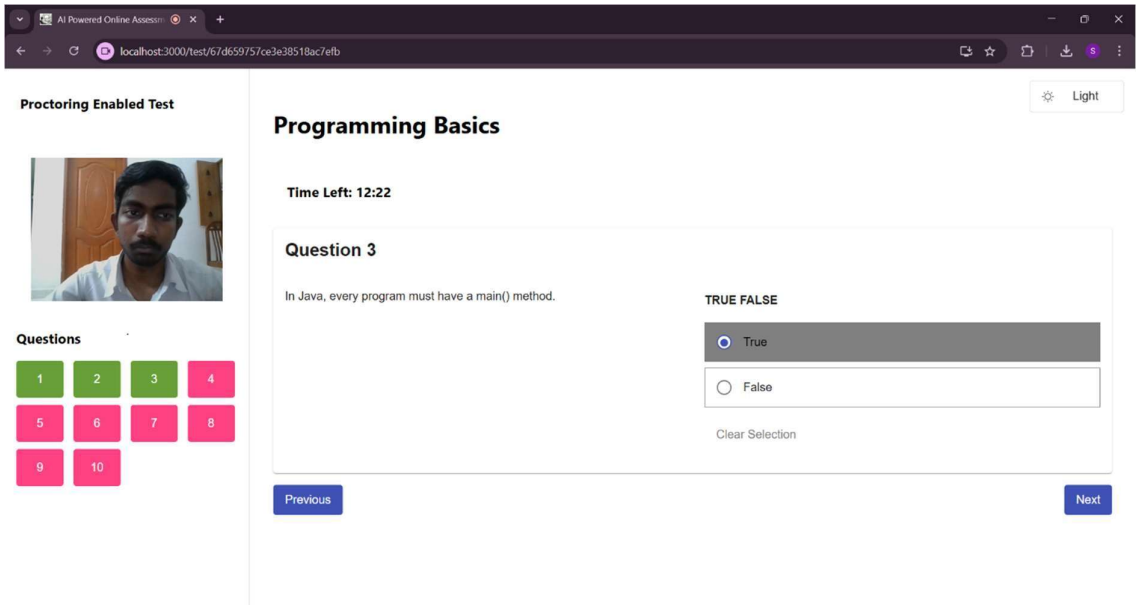

Fig: 9.1.28 True or False question type

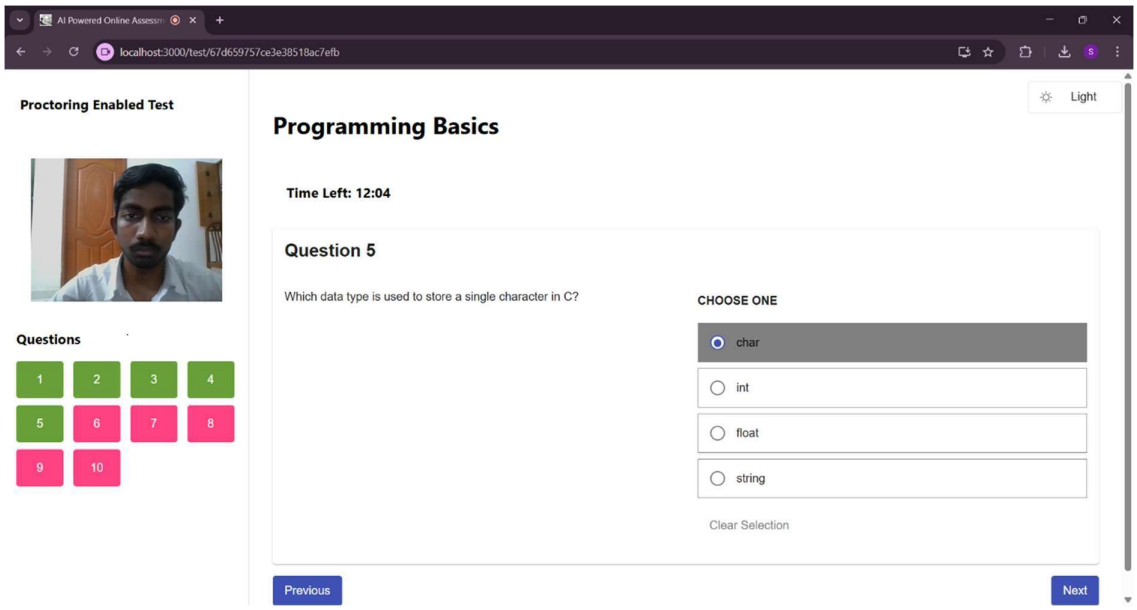

Fig: 9.1.29 Choose one question type

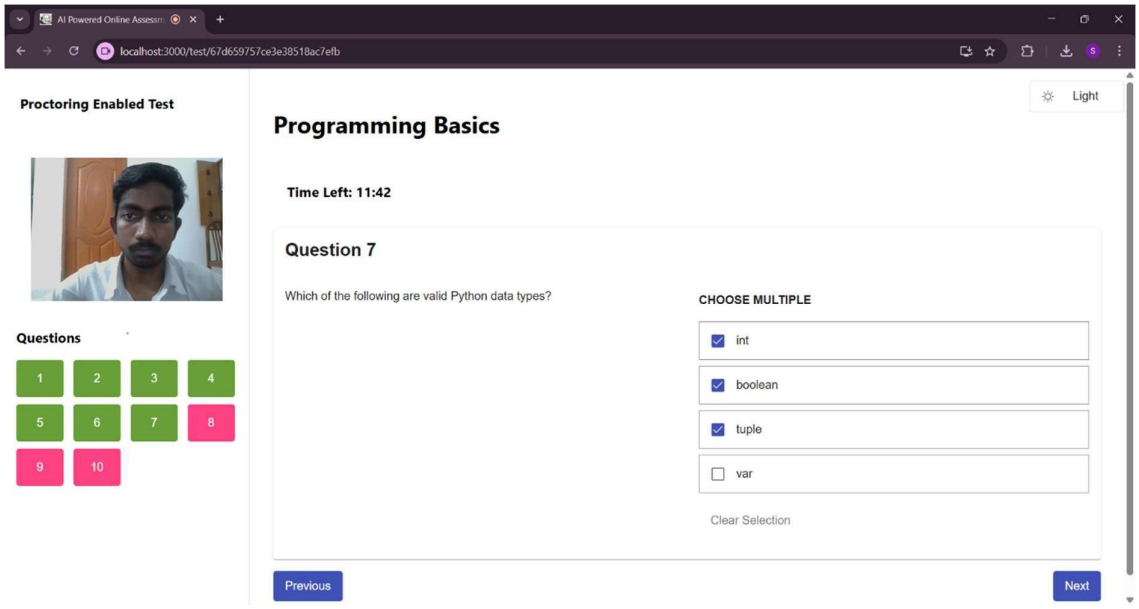

Fig: 9.1.30 Choose Multiple question type

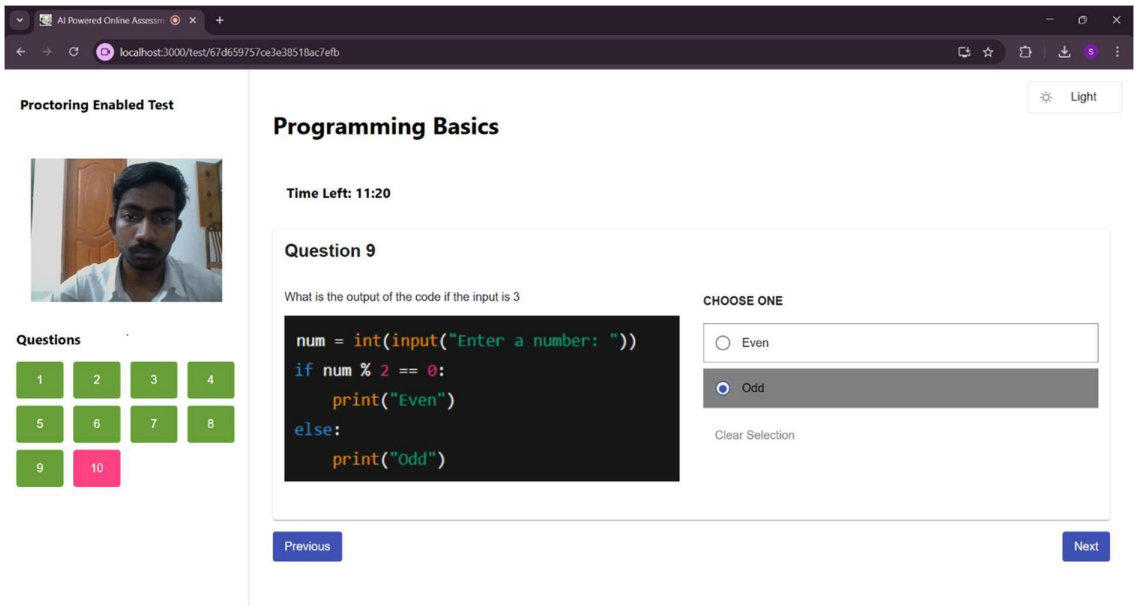

Fig: 9.1.31 Image based Question

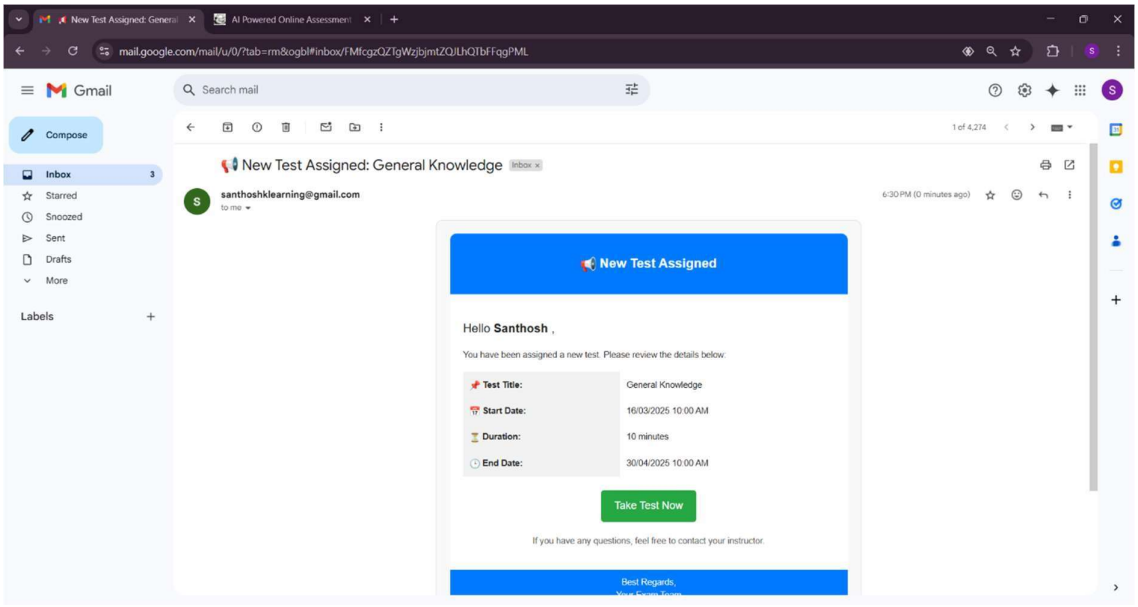

Fig: 9.1.32 Student's Mail for test assigned

9.2 USER MANUAL

The user manual contains all information for the user to make full use of the system information. This manual includes a description of the system functions and capabilities. The step-by-step procedures for the system access and functional inputs are described in this manual for future use.

The AI-Powered Online Assessment Proctoring System is designed to facilitate seamless and secure online assessments for students, with enhanced monitoring features for teachers and admins. This manual provides instructions on how different users—Admin, Teacher, and Student—can interact with the system.

User Roles and Functionalities

Admin

Admins have the ability to:

- Create students either individually or in bulk via an Excel file.
- View student details.

- Delete students from the system.
- View reports of all tests by selecting a test from a dropdown and clicking "Get Reports."
- Download test reports in Excel format.

Teacher

Teachers can:

- Create new tests and edit existing ones.
- Define test questions under four types:
    - Fill in the Blanks
    - True or False
    - Choose One
    - Choose Multiple
- Upload images for questions (optional).
- Assign tests to students.
- View test reports for the tests they created.

Student

Students can:

- Attend assigned tests.
- View reports of completed tests.
- Download test reports.

User Login & Profile Management

- All users log in through a common login page by entering their email and password.
- Based on their role (Admin, Teacher, or Student), they are redirected to the respective dashboard.

- If a user forgets their password, they can click "Forgot Password," enter their email, and receive a password reset link via email.
- Users can also update their profile and reset their password from the profile page.

Test Management & Proctoring

- Teachers create and assign tests with different question types.
- Students take the test in a real-time proctoring environment.
- The system monitors students and alerts them with toast messages for suspicious activities.
- Alerts are saved and included in the final report.
- After submitting the test, reports are auto-generated and accessible in the "View Reports" section.

Test Reports

- Admins can generate reports for all tests.
- Teachers can view and analyze reports for the tests they created.
- Students can access and download reports of their completed tests.

9.3 CONCLUSION

The AI-Powered Online Assessment Proctoring System revolutionizes the way online exams are conducted, ensuring integrity, security, and fairness. By leveraging AI-driven proctoring mechanisms, the system enhances the credibility of online assessments through real-time monitoring, face detection, and behavior analysis. It provides a seamless and user-friendly experience for administrators, teachers, and students, making test management efficient and reliable.

With features such as automated test creation, multiple question formats, detailed reporting, and real-time alerts for suspicious activities, this system meets the evolving demands of digital education. As technology continues to advance, this solution demonstrates the transformative power of AI in academic assessments, paving the way for a secure and scalable future in online learning. It not only improves test security but also fosters transparency, making it an essential tool for modern educational institutions.

CHAPTER X

REFERENCES

[1] Chaudhary, P., & Goyal, R. (2021). "AI-Based Proctoring Systems for Online Examinations: Challenges and Opportunities." International Journal of Emerging Technologies in Learning (iJET), Vol. 16 No. 12, pp. 85-102.

[2] Sharma, R., & Gupta, V. (2020). "Ensuring Academic Integrity in Online Exams: A Comparative Analysis of AI-Powered Proctoring Tools." Journal of Educational Technology & Society, Vol. 23 No. 4, pp. 56-72.

[3] Patel, K., & Desai, R. (2022). "Machine Learning Techniques for Automated Proctoring in E-Learning Platforms." International Journal of Computer Applications, Vol. 179 No. 3, pp. 45-59.

[4] Zhang, Y., & Li, X. (2019). "Real-Time Face Detection and Behavior Analysis for Online Examination Systems." IEEE Transactions on Learning Technologies, Vol. 12 No. 1, pp. 22-34.

[5] Kumar, S., & Reddy, P. (2023). "Enhancing Online Exam Security Using AI-Driven Proctoring and Behavioral Analytics." *International Journal of Artificial Intelligence in Education*, Vol. 31 No. 2, pp. 78-95.