**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.

**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.

**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**

**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.

**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.

**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.

**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.

Top of Form

**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(8th 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 |

**Hardware Requirements**

**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 |

**Software Requirements**

**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**

**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:**

1. Conceptual Data Model
2. Physical Data Model
3. 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.

**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

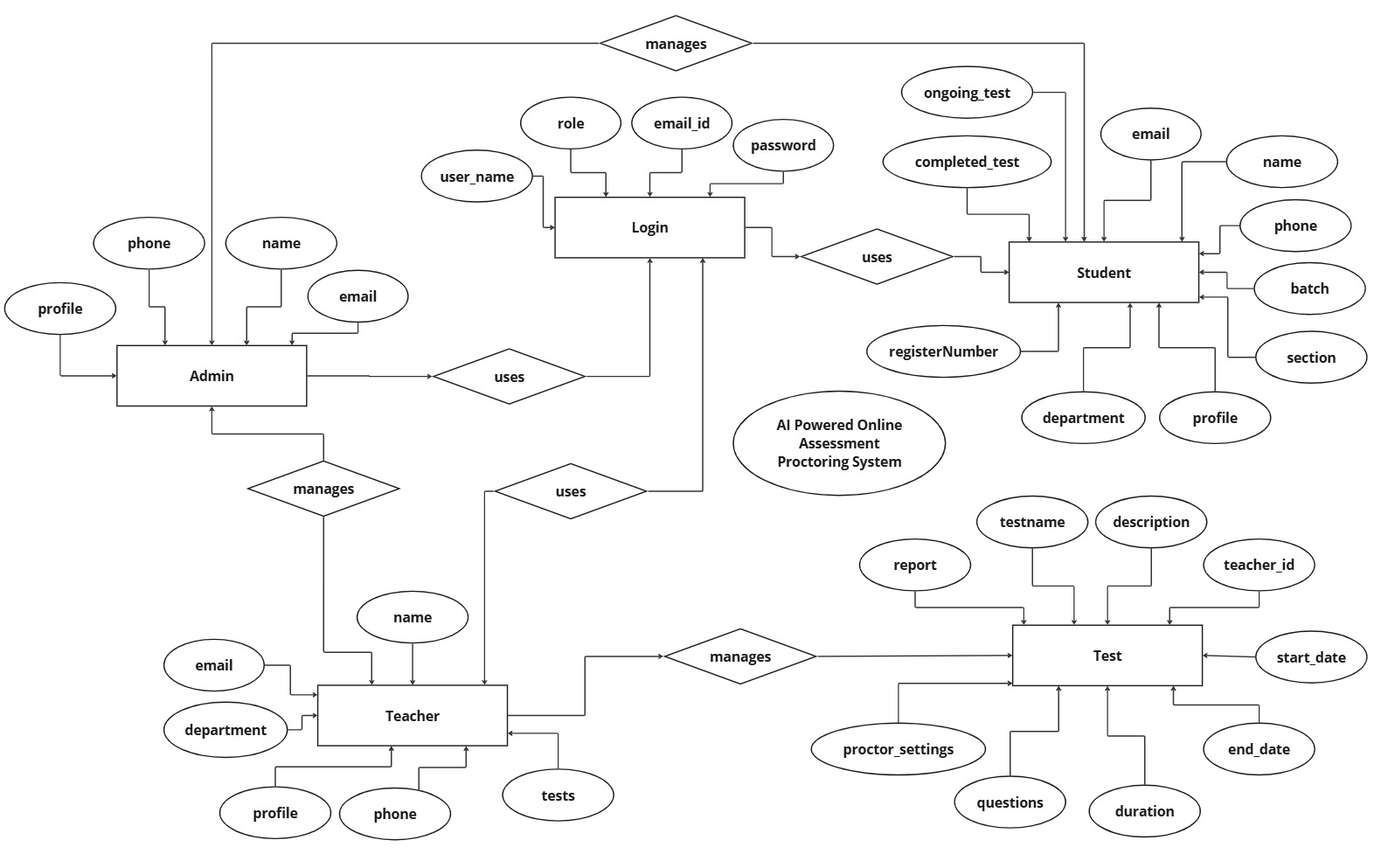
**ENTITY RELATIONSHIP DIAGRAM:**

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.



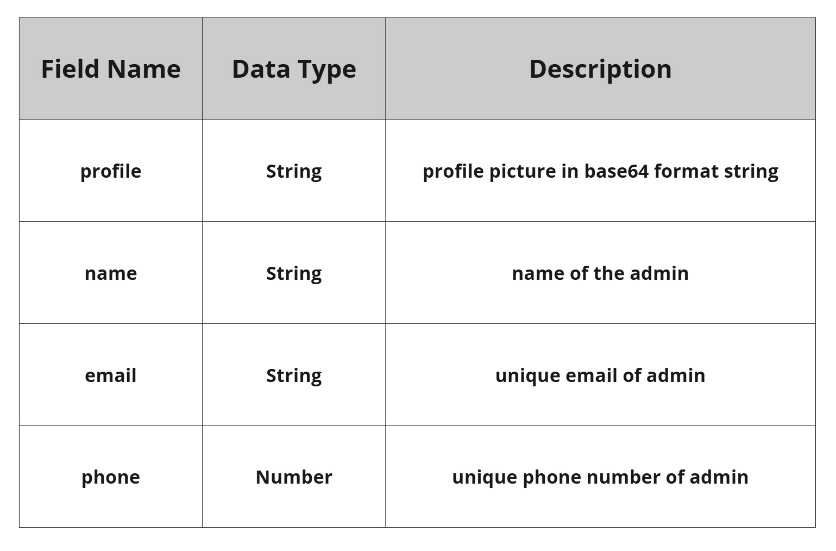
**ER Diagram**

**DATA DICTIONARY:**

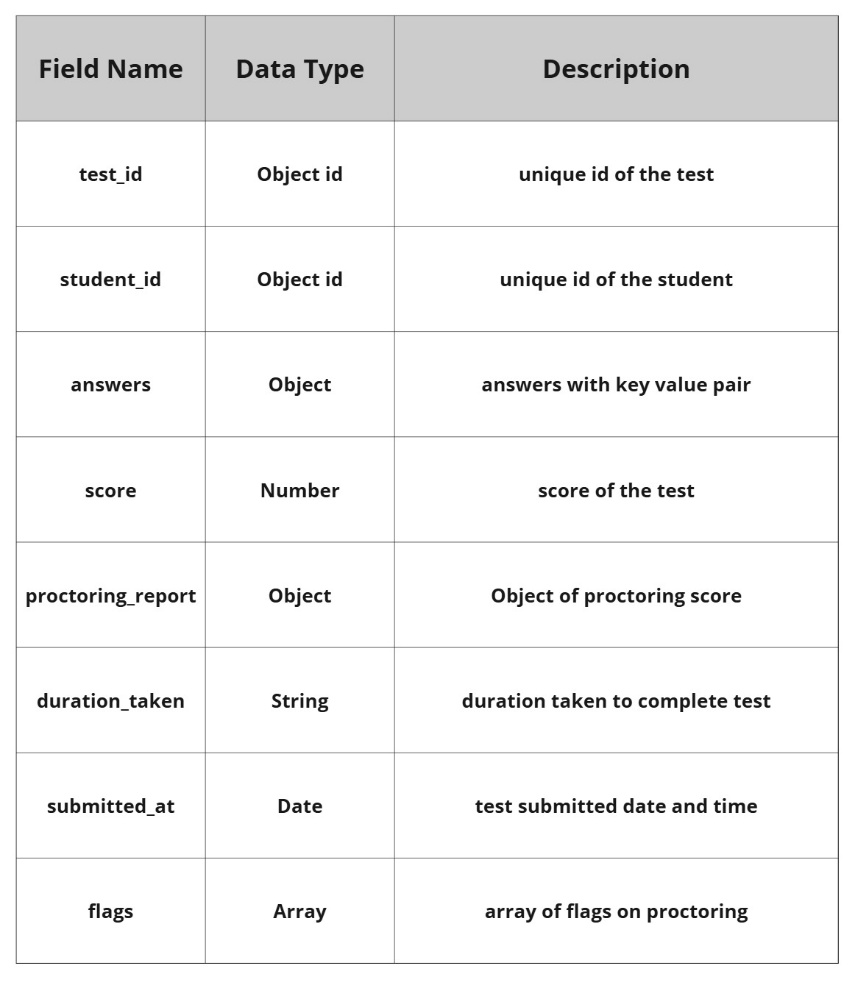
A data dictionary is a collection of metadata that describes the structure, attributes, and constraints of data elements used in a system. It provides detailed information about each data element, including its name, data type, size, allowed values, relationships, and purpose within the system. The data 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.

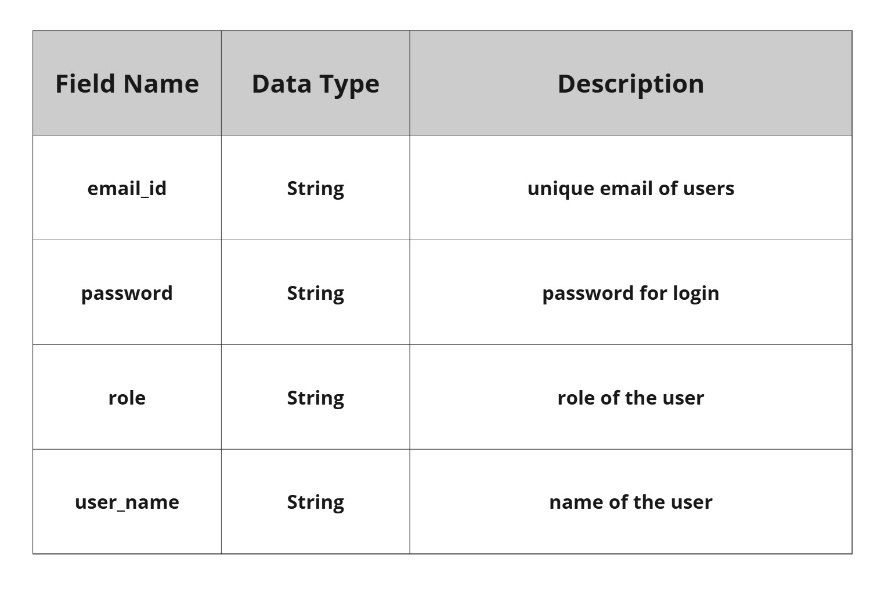
A data dictionary typically includes elements such as tables, attributes, primary keys, foreign keys, and data formats. It plays a crucial role in database design, system documentation, and application development. By providing standardized definitions, it helps prevent data redundancy, ensures consistency,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



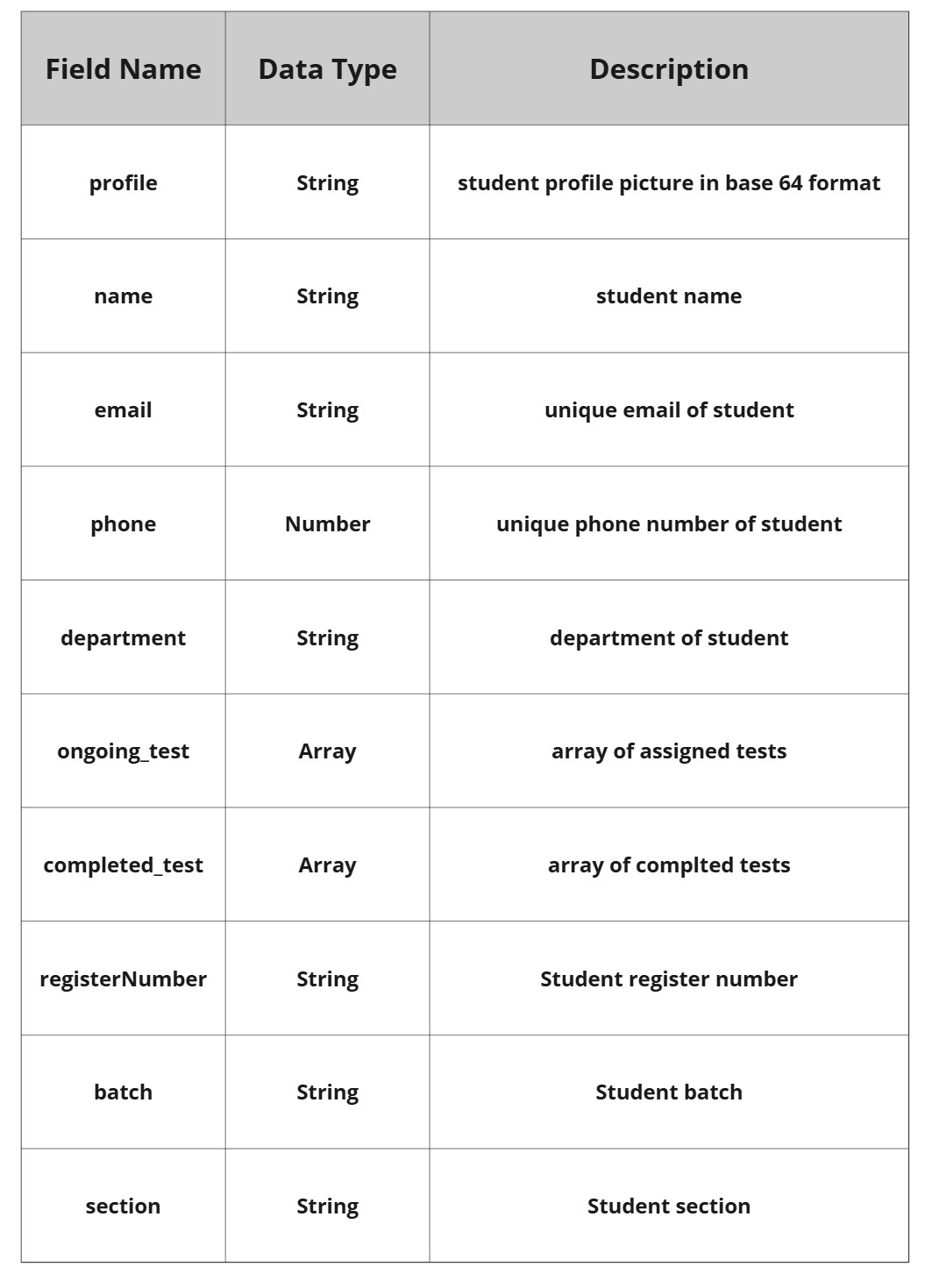
**Admin**



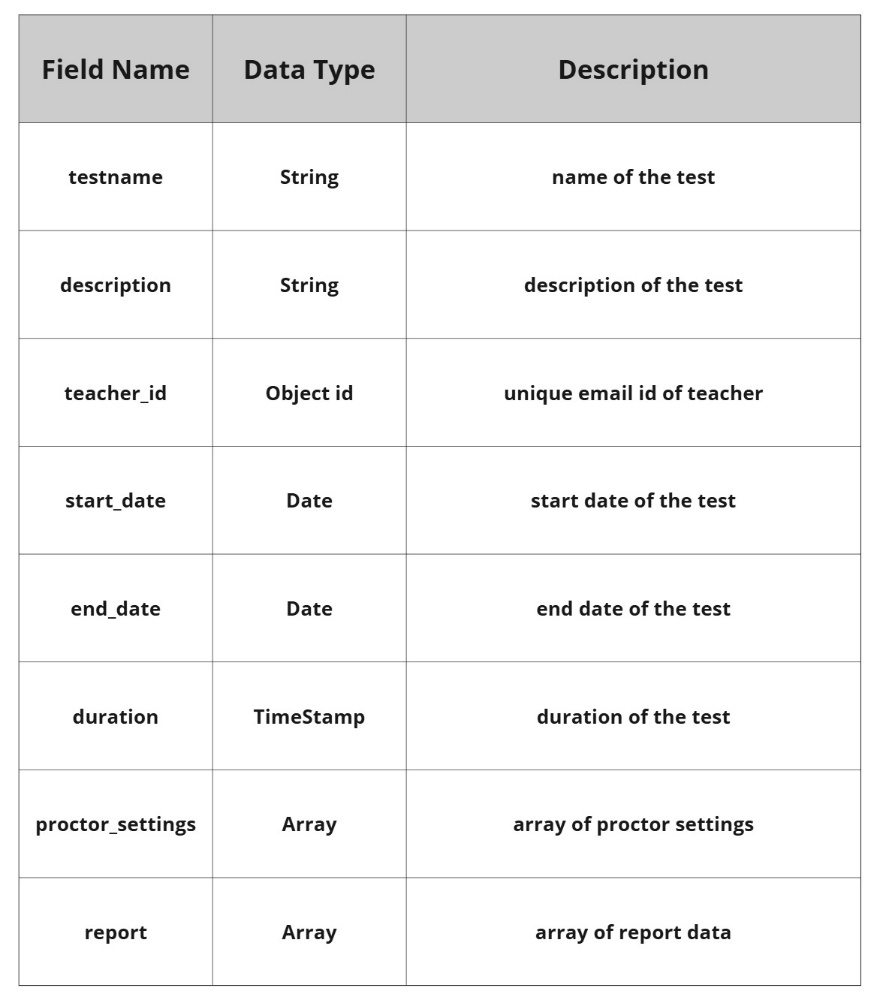
**Report**

****

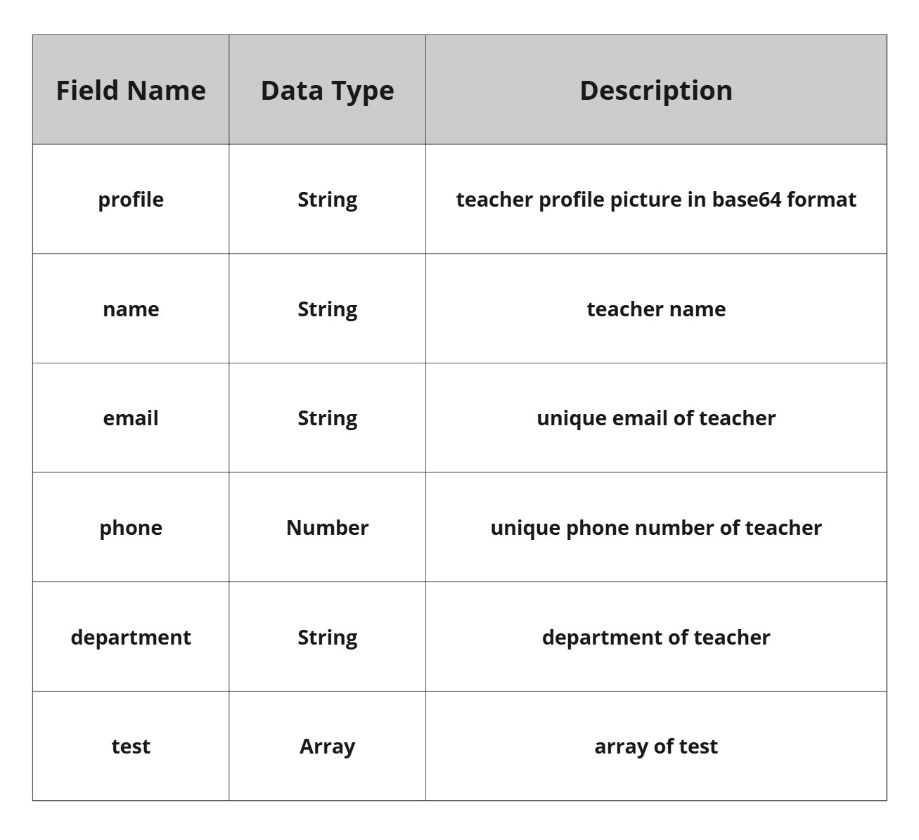
**Login**

****

**Student**

****

**Test**

****

**Teacher**

**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.

**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

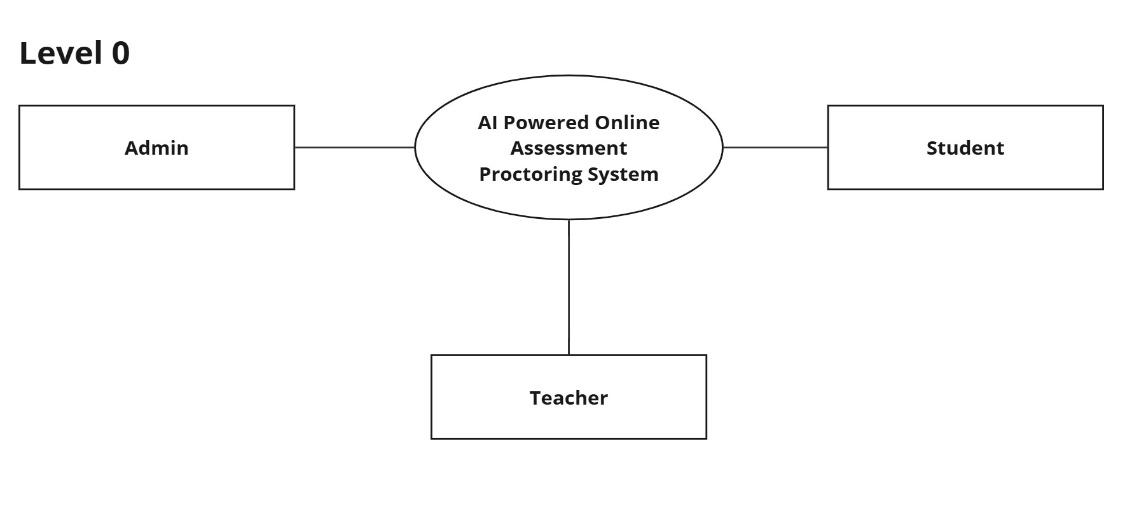
* This level decomposes the single process from the context diagram into subprocesses or functions.
* It provides an overview of the major processes within the system and how they interact.
* External entities may still be present, but the focus is on the internal processes and data flow among them.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

Level 2 DFD

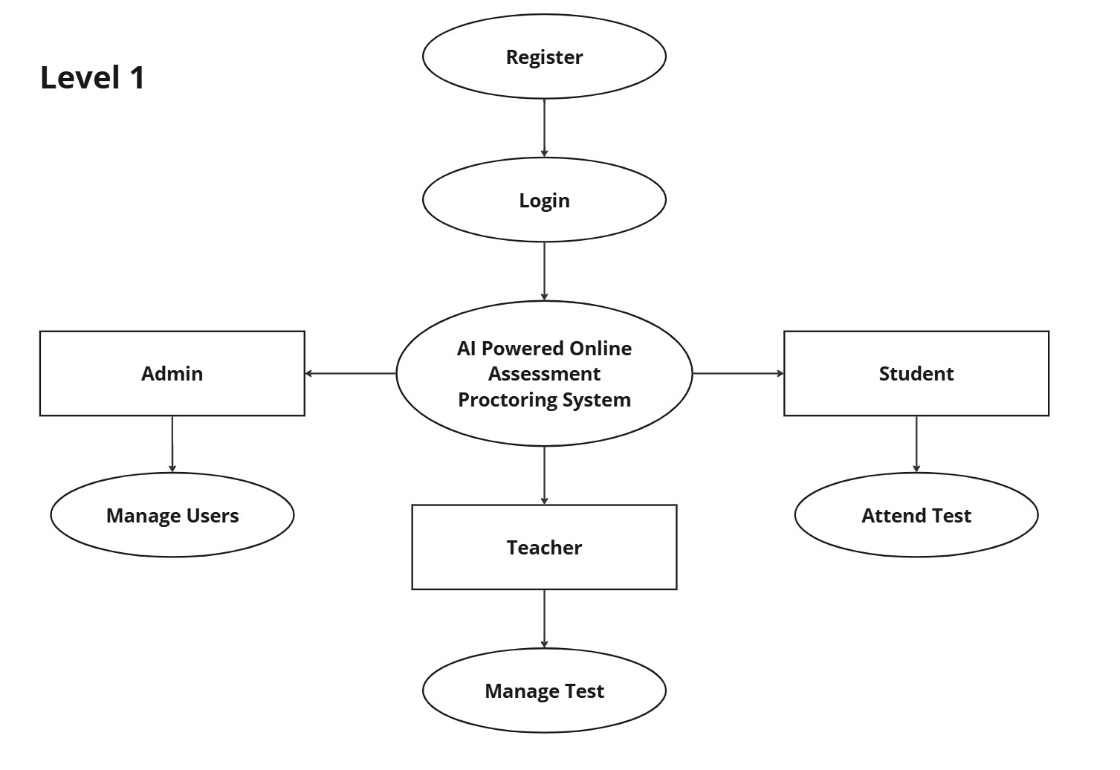
* This level further decomposes the processes from Level 1 into more detailed sub- processes.
* It provides a detailed view of how data is processed within each of the Level 1 processes.
* Data stores (where data is stored) and data transformations are often represented at this level.
* Arrows represent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

**Components of Data Flow Diagram:**

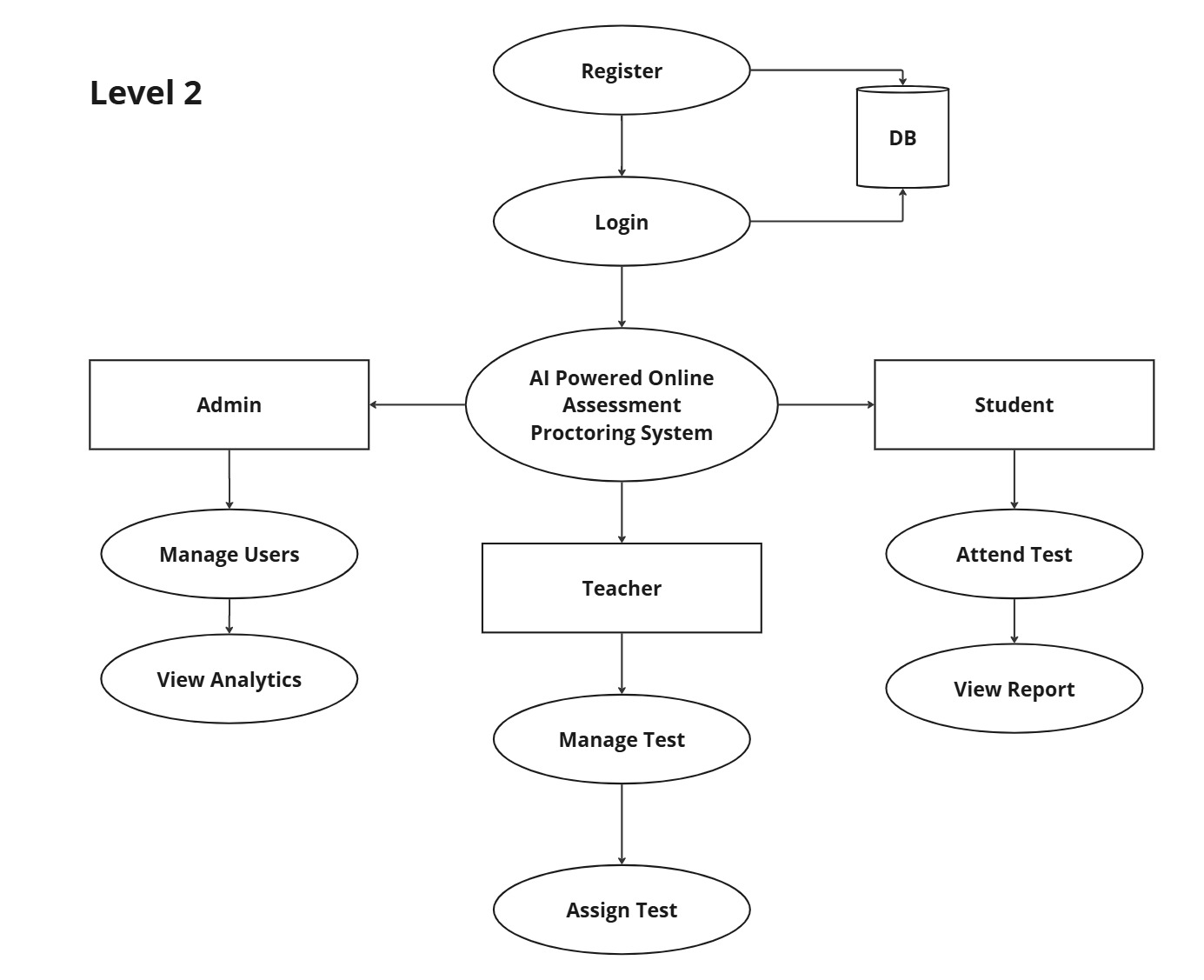
The Data Flow Diagrams includes four main component elements. They are entity, process, data store and data flow. External Entity also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed.



**DFD Level 0**

****

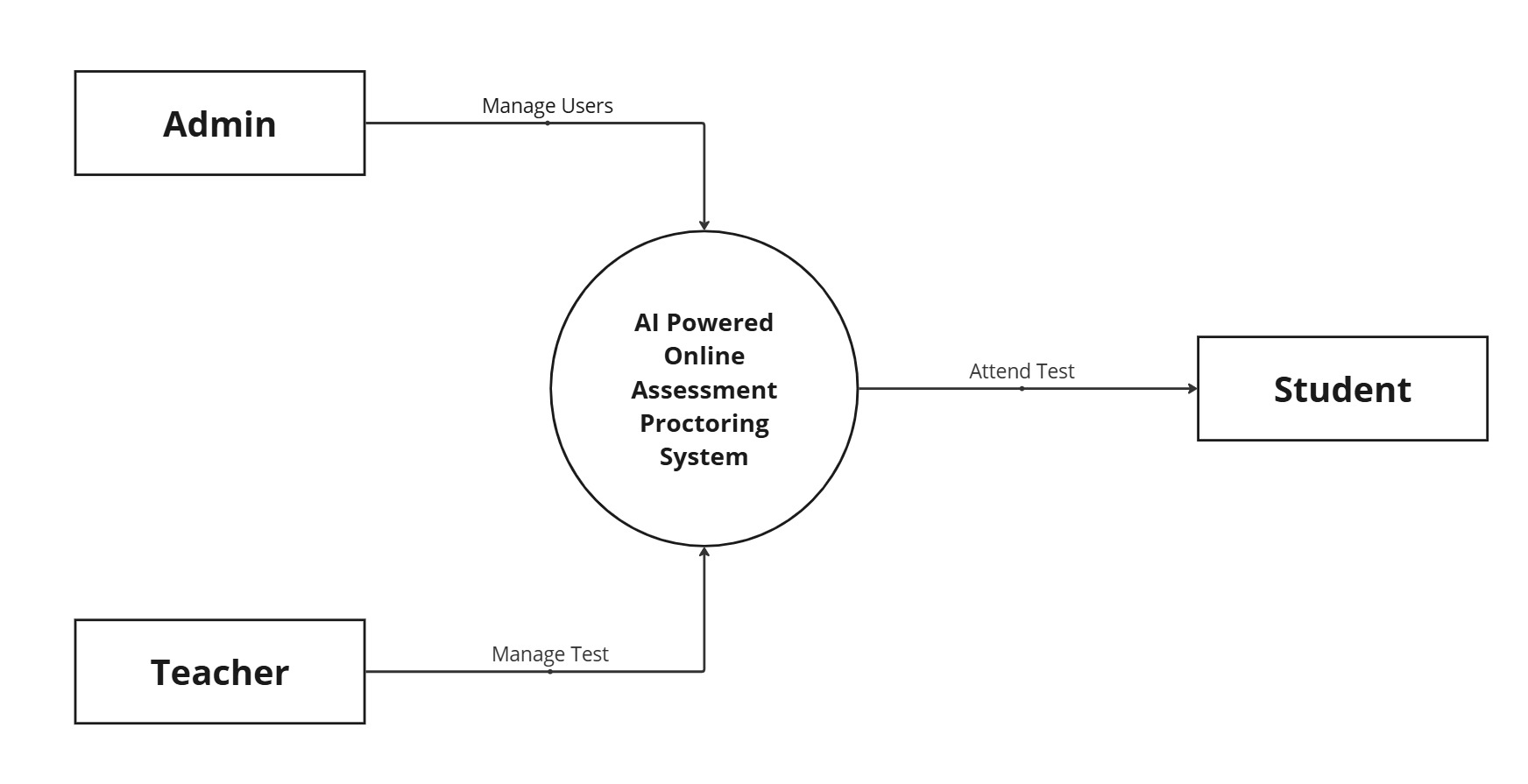
**DFD Level 1**

****

**DFD Level 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.



**Context anaylsis diagram**