**A**

PROJECT REPORT ON

**DETECTING FRAUDULENT STUDENT COMMUNICATION IN A MULTIPLE CHOICE ONLINE TEST ENVIRONMENT**

Submitted in partial fulfillment of the

requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**By**

**Ms. GIREESHA CHETLAPALLI**

**(Regd.No:21701A3308)**

Under the Guidance of

**Mr. S. SUREKHA**

###### Assistant professor



**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

ANNAMACHARYA INSTITUTE OF TECHNOLOGY AND SCIENCES

###### (AUTONOMOUS)

NEW BOYANAPALLI-516126, RAJAMPET (A.P)

(Accredited A-Grade by NAAC, Bangalore)

(Approved by A.I.C.T.E, New Delhi &Affiliated to J.N.T.U.A, ANANTHAPURAMU)

**2021-2025\*\*\***

###### ANNAMACHARYA INSTITUTE OF TECHNOLOGY AND SCIENCES (AUTONOMOUS) NEW BOYANAPALLI, RAJAMPET-516126



###### Affiliated to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR,

ANANTHAPURAMU

**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

**CERTIFICATE**

This is to certify that the project work entitled **“DETECTING FRAUDULENT STUDENT COMMUNICATION IN A MULTIPLE CHOICE ONLINE TEST**

**ENVIRONMENT”** is the bonafide work carried out by **Ms. GIREESHA CHETLAPALLI, Regd.No:21701A3308** is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor Of Technology during the year 2021-2025.

**Guide Head of the Department**

**Mrs.S.Surekha Dr.T.Hari Krishna**

**Asst professor Professor and Head**

**Department of AI&ML Department of AI&ML**

**ABSTRACT**

**ABSTRACT**

Online evaluation systems, pervasive nowadays, are known to be susceptible to higher fraud risks. This work proposes a novel and robust method to detect potential fraud acts in online multiple-choice question (MCQ) exams. For the first time, the communication probability between the examinees is statistically assessed based on the concordance of responses and answer time against null expectations and is subsequently used to identify potential fraud behavior. The model is sensitive to the direction of communication acts, distinguishing content consumption from production, as well as multiwise communication channels. Online remote tests from engineering courses at Technicon Lisboa are used as a case study. We show that the cumulative contribution of concordant responses between students, when recurrent, offers a way of signaling fraud behavior. Separating content production from consumption reveals the underlying student role played in potential fraud acts. Collusion behavior is assessed against null models of fraud and conformity, and therefore being statistically framed and offering a solid criterion to guide tutors in ascertaining fraud and discouraging communication.

**CONTENTS**

**Topics Page No.**

[1. Introduction 1](#_Toc119920)

[1.1. Purpose 2](#_Toc119921)

[1.2. Scope 2](#_Toc119922)

[1.3. Need for System 2](#_Toc119923)

[1.3.1 Existing System 3](#_Toc119924)

[1.3.2 Proposed System 3](#_Toc119925)

[1.4. System Architecture 5](#_Toc119926)

[2. Software Requirement Analysis And Specifications 6](#_Toc119927)

[2.1. Product Perspective 6](#_Toc119928)

[2.2. Product Functions 6](#_Toc119929)

[2.3. User Characteristics 7](#_Toc119930)

[2.4. Constrains 8](#_Toc119931)

[2.5. Modules 9](#_Toc119932)

[2.6. Functional And Non-Functional Requirements 10](#_Toc119933)

[2.6.1 Functional Requirements 10](#_Toc119934)

[2.6.2 Non-Functional Requirements 10](#_Toc119935)

[2.7. Specific Requirements 11](#_Toc119936)

[2.7.1 Software Requirements 11](#_Toc119937)

[2.7.2 Hardware Requirements 12](#_Toc119938)

[2.8. Software Development Life Cycle 12](#_Toc119939)

[2.8.1 Waterfall Model 13](#_Toc119940)

[2.9. System Study 14](#_Toc119941)

[2.10. Technologies Used 15](#_Toc119942)

[3. System Design 26](#_Toc119943)

[3.1. Database Design (ER-Diagram) 26](#_Toc119944)

[3.1.1 ER-Model 26](#_Toc119945)

[3.2. Data Dictionary 33](#_Toc119946)

[3.3. Normalization 35](#_Toc119947)

[3.4. UML Diagrams 39](#_Toc119948)

[4. Testing 57](#_Toc119949)

[4.1. Introduction 57](#_Toc119950)

[4.2.Testing Methodologies 59](#_Toc119951)

[4.2.1 Unit Testing 59](#_Toc119952)

[4.2.2 Integration Testing 60](#_Toc119953)

[4.2.3 Client Acceptance Testing 60](#_Toc119954)

[4.2.4 Output Testing 61](#_Toc119955)

[4.2.5 Approval Checking 61](#_Toc119956)

[4.3. Software Testing Strategy 61](#_Toc119957)

[4.3.1 Framework Testing 62](#_Toc119958)

[4.4. Test cases 63](#_Toc119959)

[5. Implementation 64](#_Toc119960)

[5.1. Sample Screens 64](#_Toc119961)

###### Conclusion Bibliography Appendix – A

* URL Listings
* References

**Appendix – B**

* Glossary

###### Appendix – C

* List of Figures
* List of Tables
* List of Screen

**Appendix – D**

* Base Paper

**INTRODUCTION**

Detecting Fraudulent Student Communication In

A Multiple Choice Online Test Environment

# 1. INTRODUCTION

Over the last decade, advancements in technology have offered students the opportunity to enroll in a wide array of online courses, alongside traditional face-toface classes. Online courses have gained popularity due to their flexibility, allowing students to work at their own pace and reduce attendance costs. University administrators have embraced online content and assessments to expand student access, a trend accelerated by the COVID-19 pandemic, which necessitated remote teaching. However, remote teaching poses challenges, particularly in the realm of unsupervised online testing, which has been associated with increased fraud risks, challenging the principles of fair evaluation. Fraud encompasses various illegitimate activities aimed at improving assessment performance, including the use of unauthorized materials, copying, collusion among examinees, obtaining test contents in advance (pre-knowledge), impersonation, or external assistance during the test. This study focuses on collusion among examinees, a common form of online cheating in multiple-choice question (MCQ) exams conducted remotely. Existing methods for detecting collusion in online exams suffer from several limitations

* They assume fixed question orders and reversible answering.
* They overlook the distinct roles and cumulative effects of advertent content sharing across multiple examinees using unauthorized communication platforms.
* They fail to reliably assess deviations in behavioral statistics against expected In norms.
* To address these limitations comprehensively, this study introduces a novel.
* Method for detecting potential fraud acts based on timestamped answer records from online quizzes with shuffled questions. This method captures potential instance of information exchange among examinees taking the test simultaneously. In the context of multiple-choice online tests, this approach is increasingly necessary due to the possibility of fraud through direct in-room communication or electronic messaging platforms like WhatsApp and

Messenger.

* Handling collusion, regardless of the communication method used, is the central focus of this study, aiming to enhance the integrity and fairness of online assessments in educational setting.

## 1.1 Purpose

Online evaluation systems have become pervasive in modern education, yet they are increasingly vulnerable to fraudulent activities. This work introduces a novel and robust method aimed at detecting potential fraud in online multiple-choice question (MCQ) exams. Unlike previous approaches, our method statistically evaluates the communication probability between examinees based on the concordance of responses and answer time against null expectations. This assessment is instrumental in identifying potential fraudulent behavior among examinees.

## 1.2 Scope

Our system focuses specifically on detecting fraudulent communication in online multiple-choice tests. It employs sophisticated algorithms and statistical analysis techniques to assess the communication patterns between students during such tests. By analyzing the concordance of responses and answer times, our system can identify suspicious behaviors indicative of collusion or cheating among students. Furthermore, our system is designed to be adaptable across various educational institutions and online testing platforms. It can be integrated seamlessly into existing online evaluation systems, providing an additional layer of security and integrity to the assessment process.

## 1.3 Need For System

The need for our system arises from the growing reliance on online evaluation systems in education, particularly in the context of remote learning and digital assessments. As more educational institutions transition to online platforms for conducting tests and exams, there's a pressing need to safeguard the integrity and reliability of these assessments.

Therefore, there's a critical need for effective fraud detection mechanisms that can identify and deter such behaviors proactively. Our system addresses this need by offering a proactive approach to detecting fraudulent student communication, thereby ensuring the integrity and fairness of online assessments. By leveraging advanced statistical analysis and data-driven insights, our system empowers educators and institutions to maintain academic standards and uphold the value of online education.

#### 1.3.1 Existing System

Previous research suggests that students cheat for various reasons, which extend beyond the realm of online testing. These reasons include factors such as low grades and ineffective study strategies, poor time management skills, personal values and attitudes towards achievement, fear of punishment, class attendance, peer pressure, extrinsic versus intrinsic motivations to learn, and age. Ladyshewsky observed that certain student profiles may attempt to cheat regardless of the instructional mode. While earlier studies did not conclusively demonstrate that remote online assessments increase the likelihood of cheating, recent findings indicate a significant rise in dishonest behavior in remote assessments.

Disadvantages

* Assume fixed question orders and reversible answering.
* Neglect the distinguished roles and multi wise cumulative effects from inadvertent

content sharing exerted in unauthorized communication platforms.

* Do not reliably test the deviation of the acquired behavioral statistics

againstplausible expectations.

#### 1.3.2 Proposed System

The proposed methodology outlined above offers a comprehensive approach to evaluating fraudulent communication in online testing environments. It begins with a meticulous pre-analysis of data to accommodate diverse patterns of fraudulent behavior.

Firstly, a robust statistical framework is established to assess the probability of pairwise student communication. This framework considers various factors such as matched answers, choice probability, response times (directionality), and the recurrence of suspicious behavior.

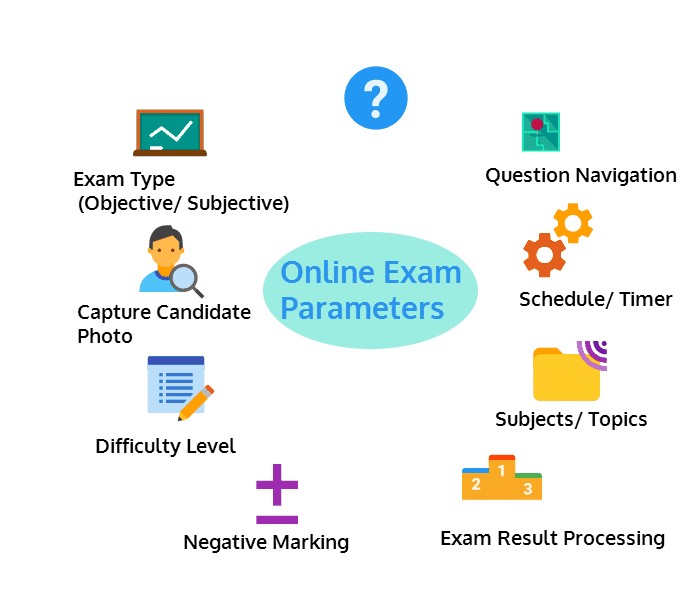
Secondly, the methodology employs a network representation to visualize potential communication acts among students. This representation enables the examination of directional multi wise communication acts within the context of a multiple-choice online test. Thirdly, null models of compliance and fraud are developed based on a principled understanding of inadvertent communication. These models serve as benchmarks to test collusion dynamics and identify fraudulent behavior with high statistical significance.

Finally, scoring, clustering, and visualization techniques are utilized to facilitate the interpretation of inadvertent communication pathways. This enables the generation of actionable recommendations and supports course tutors in conducting further investigations and implementing preventive measures.

Advantages

* **Accessibility:** Online testing platforms provide convenient access to assessments for students regardless of their location, allowing for flexible scheduling and accommodating remote learners.
* **Cost-Effectiveness:** Online testing reduces the need for physical test administration facilities, paper-based materials, and manual grading, resulting in cost saving for educational institutions.
* **Scalability:** Online testing platforms can accommodate a large number of classes or even massive open online courses (MOOCs).
* **Automation:** Automated features such as question randomization, time limits and plagiarism detection streamline the test administration process and reduce the administrative burden on instructors.
* **Data Analysis:** Online testing platforms collect valuable data on student performance, including response times, question difficulty, and completion rates which can be analyzed to inform instructional design and assessment strategies.

## 1.4 System Architecture



**Fig.no 1.4 System Architecture**

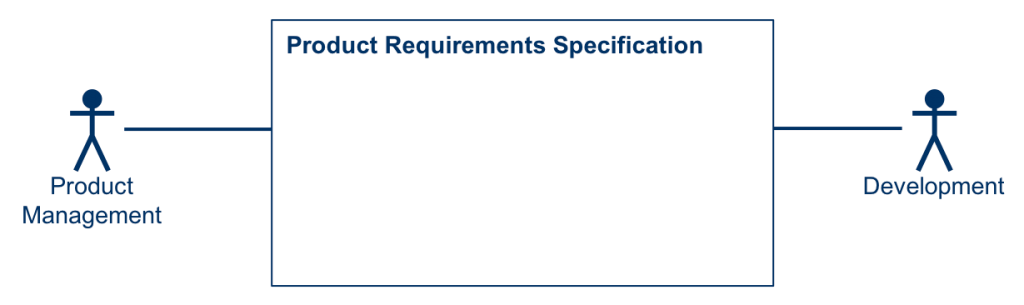
**SOFTWARE REQUIREMENT ANALYSIS AND**

**SPECIFICATIONS**

# 2. SOFTWARE REQUIREMENT ANALYSIS AND SPECIFICATIONS

### 2.1 Product Perspective

An essential aspect necessitates assurance, which involves bridging the gap between product management and development. It entails thoroughly defining a product in terms of its associated requirements, encompassing all the necessary specifications that must be explicitly described and continuously available.



**Fig.no 2.1 Product Perspective**

### 2.2 Product Functions

In product management, the product requirements specification serves as the central tool, fulfilling the following purposes

1. It articulates how product management addresses partner needs and requests.
2. It communicates to development teams what the product or new features to be developed will entail.
3. The accompanying figure illustrates these relationships, focusing on the product perspective.
4. Product management is responsible for defining a product perspective that aligns with the expectations and requirements of the product's stakeholders. Additionally, it should consider and serve relevant relationships with adjacent systems.

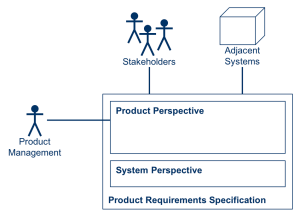


Fig.no 2.2 Characteristics of product perspective

The key characteristics of the product perspective include

* Describes the external view of the product.
* Answers the question "What".
* Represents what the product is.
* Describes what the product does.
* Addresses both the business and usage aspects of the product.
* Utilizes problem domain language and concepts.
* Presents a "Black Box" view, focusing on the external behavior and functionality of the product without delving into its internal workings.

### 2.3 User Characteristics

Definition

This section outlines the characteristics of users in three distinct contexts the end user who interacts with the machine translation system; the end user of the final product of the translation process, which may include post-editing the organization producing the machine translation system. Relevant stakeholders may include

* Interpreters
* Post-editors
* Translation buyers
* Higher-level

### 2.4 Constrains

Pragmatic requirements are product features or functionalities that developers must implement to enable users to accomplish their tasks. It is essential to clearly define them for both the development team and the stakeholders. Typically, pragmatic requirements describe system behavior under unambiguous conditions. For example

* The system sends a confirmation request after the user enters personal information.
* A search feature allows a user to search among multiple orders to allocate a provided receipt.
* The system sends a confirmation email when a new user account is created.
* Types of functional requirements and their specifics.

Functional requirements can be categorized by various criteria. For instance, they can be grouped based on the roles a given component should perform in the final product. Naturally, these requirements may vary depending product under development, but examples of functional requirements might include

* Authentication
* Authorization levels
* Compliance with regulations or standards
* External interfaces
* Transaction processing
* Reporting
* Business rules, etc.

### 2.5 MODULES

**Admin Module:** in this Module, The admin use his credentials like his/her UserId and Password for login. After successful login to admin menu, can perform different activities and they are

* Viewing all users, all registered user can be view by admin
* View all dataset, all dataset that has upload can view
* View detecting student communication type i.e., how many Genuine and Fraud a

Are there

* View all communication type result, by selecting the type it will show that data set
* Logout

**Registration Module:** before going to update the dataset by the user he must register and user should fulfill the detail form like

* Username
* Password
* Email
* Mobile
* Gender
* Address
* Pin code

**User Module:** in this Module, after successful registration by the user he can user his credentials i.e., User Id and Password and user can have different activities

* My profile – in which he/her profile visualize and can edit
* Upload dataset, here our dataset can be upload for test
* View uploaded dataset, here we can see the uploaded dataset
* Find test communication type, here we need to enter std-Id,Std –Name and exam Mode

* Find test communication type by hash code, here can select which category of result we have use i.e., based on Guanine or Fraudulent
* Logout

### 2.6 Functional And Non-Functional Requirements

#### 2.6.1 Functional Requirements

* Functional requirements delineate the intended actions of the system and can be segmented into various categories.
* Inputs: Specifies the data the system should accept.
* Outputs: Defines the results or data the system should generate.
* Data Storage: Determines the data that must be stored by the system.

**Computations:** Describes the calculations or processes the system needs to perform.

Input Design

Input design facilitates interaction between the information system and users, establishing processes for data preparation and entry. It involves methods to transform transaction data into usable formats, such as reading from documents or direct entry. Input design aims to streamline processes, minimize errors, and ensure user security and privacy. Key considerations include

* Defining input data requirements.
* Organizing and coding data.
* Providing user guidance.
* Implementing input validations and error handling.

#### 2.6.2 Non-functional Requirement

Non-functional requirements (NFRs) address fundamental aspects of software systems, focusing on qualities beyond specific functionalities. Failure to address NFRs adequately can result in user dissatisfaction, software conflicts, and increased time and cost to rectify issues.

**Types of Non-functional Requirements**

* **Scalability:** The system's ability to handle increasing loads without compromising performance.
* **Reliability:** The system's ability to perform consistently and predictably under various conditions.
* **Regulatory Compliance:** Ensuring the system adheres to relevant regulations, standards.
* **Maintainability:** Ease of maintaining and updating the system over time.
* **Serviceability:** The ease with which the system can be repaired or serviced.
* **Utility:** The usefulness and value provided by the system to users.
* **Availability:** Ensuring the system is accessible and operational when needed.
* **Usability:** The system's ease of use and user experience.
* **Interoperability:** The system's ability to interact and operate with other systems  or components.
* **Environmental Considerations:**  Factors related to the environmental impact of the system, such as energy consumption or sustainability.

Addressing non-functional requirements is crucial for ensuring the overall success and effectiveness of a software system. Each requirement category contributes to the system's overall performance, reliability, and user satisfaction, making them essential considerations throughout the development lifecycle.

### 2.7. Specific Requirements

#### 2.7.1. Software Requirements

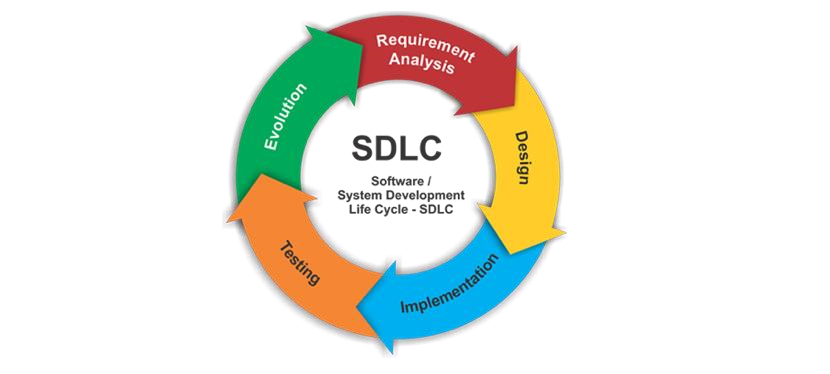
|  |  |  |  |
| --- | --- | --- | --- |
| • | Operating system |  | : Windows 10 or above |
| • | Coding Language |  | : Java |
| • | Front-End |  | : JSP |
| • | Back-End |  | : MySQL |
| • | Designing |  | : Html, CSS, JavaScript. |

#### 2.7.2. Hardware Requirements

* Processor : Intel i3 processor
* RAM : 4 GB(min)
* Hard Disk : 500 GB

### 2.8. Software Development Life Cycle

Software Development Life Cycle Models and Methodologies



**Fig.no 2.8.1 SDLC**

The Software Development Life Cycle (SDLC) is a series of stages that provide a structured approach to the software development process. It encompasses understanding the business requirements, eliciting needs, converting concepts into functionalities and features, and ultimately delivering a product that meets business needs. A proficient software developer should possess adequate knowledge to select the appropriate SDLC model based on project context and business requirements.

Therefore, it is essential to select the right SDLC model tailored to the specific concerns and requirements of the project to ensure its success. To explore more about choosing the right SDLC model, you can follow this link for additional information.

Further more to delve deeper into software lifecycle testing and SDLC stages, followthe highlighted links here.The exploration will cover various types of SDLC models, their benefits, disadvantages, and when to use them. SDLC models can be viewed as tools to enhance product delivery.

Therefore, understanding each model, its advantages, disadvantages, and the appropriate usage is crucial to determine which one suits the project context.

Types of Software developing life cycles (SDLC)

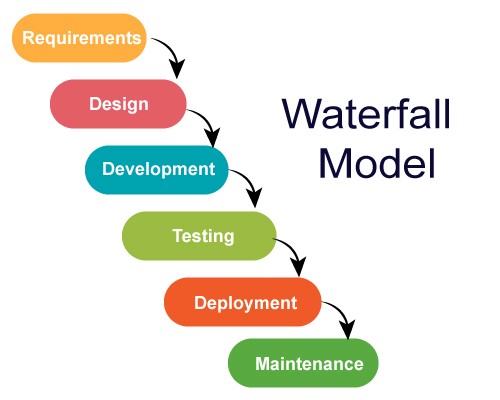
* [Waterfall Model](http://melsatar.blog/2018/02/16/the-waterfall-model-a-different-perspective/)
* [V-Shaped Model](https://melsatar.blog/2018/08/27/the-validation-and-verification-model-the-v-model/)
* [Evolutionary Prototyping](http://en.wikipedia.org/wiki/Software_prototyping) Model
* [Spiral](http://en.wikipedia.org/wiki/Spiral_model) Method ([SDM)](http://en.wikipedia.org/wiki/Software_development_methodology)
* [Iterative and Incremental](http://en.wikipedia.org/wiki/Iterative_and_incremental_development) Method
* [Agile development](http://en.wikipedia.org/wiki/Agile_software_development)

#### 2.8.1 Waterfall Model

The Waterfall Model follows a linear, sequential flow, where progress moves steadily downwards (like a waterfall) through the phases of software development. Each stage in the development cycle begins only after the previous stage is completed. The waterfall approach does not accommodate going back to a previous stage to address changes in requirements. It is the oldest and most well-known method used for software development. The five-stage waterfall model, based on Winston W. Royce's requirements, divides development processes into the following stages

1. analysis,
2. design
3. implementation
4. testing and
5. operation

The waterfall model can be broken down into multiple phases.



**Fig.no 2.8.2 water fall model**

**Advantages**

1. Simple to clarify for the clients.
2. Structures approach.
3. Stages and exercises are distinct.
4. Assists with arranging and timetable the task.

### 2.9. System Study

Feasibility Study

The feasibility study evaluates the practicality of the project and enhances fundamental understanding through a comprehensive approach. During system assessment, the integrity evaluation of the proposed structure is crucial to ensure it does not burden the organization. Three key assessments conducted during feasibility appraisal are

* Economic Feasibility
* Technical Feasibility
* Social Feasibility

**Economic Feasibility**

This study examines the financial impact of the system on the organization. It assesses the resources available for system development and justifies expenses. The design should be economically viable, leveraging freely available enhancements wherever possible, with consideration given to necessary purchases.

**Technical Feasibility**

This study assesses the specific requirements of the system and ensures it does not overly strain existing technical resources. Excessive demands on technical resources can lead to burdens on users. The design should have modest technical requirements.

**Social Feasibility**

This aspect examines the level of acceptance of the system by users. It involves establishing effective means to familiarize users with the system and ensuring they perceive it as a necessity rather than a threat. User confidence should be bolstered through clear communication and user training. Social feasibility encompasses analyzing how individuals interact within the system or organization and evaluating social impacts to understand their extent and scale. While cost-benefit analysis is a vital component of economic evaluation, broader questions should also be addressed to assess socio-economic impacts comprehensively

### 2.10. TECHNOLOGIES USED

JAVA TECHNOLOGY

Java technology serves as both a programming language and a platform, offering a versatile environment for software development. The Java Programming Language encompasses several key characteristics, including simplicity, architecture neutrality object orientation, portability,distribution capability, high performance, interpretation, multithreading support, robustness, and dynamism.

The Java Programming Language

The Java programming language is characterized by a myriad of buzzwords that underscore its versatility and effectiveness

**Simple:** Java boasts a straightforward syntax and clear structure, making it easy to learn and use.

**Architecture neutral:** Java's architecture-neutral design allows it to run on any platform without modification, ensuring compatibility across diverse environments.

**Object-oriented:** Java follows the object-oriented programming paradigm, emphasizing the creation and manipulation of objects to achieve functionality.

**Portable:** Java programs can be compiled into platform-independent bytecode, enabling them to run on any device or operating system with a Java Virtual Machine (JVM) installed.

**Distributed:** Java supports distributed computing, allowing applications to be easily distributed across multiple systems and networks.

**High performance:** Despite its platform independence, Java offers high performance through efficient bytecode execution and optimization techniques.

**Interpreted:** Java programs are compiled into bytecode by the Java compiler and then interpreted by the JVM at runtime, facilitating dynamic execution.

**Multithreaded:** Java enables concurrent programming through its built-in support for multithreading, allowing multiple tasks to run simultaneously within a single program.

**Robust:** Java prioritizes reliability and robustness by incorporating features such as strong memory management, exception handling, and type safety.

**Dynamic:** Java's dynamic nature allows for runtime adaptation and modification of program behavior, enhancing flexibility and responsiveness.

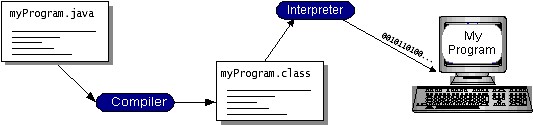
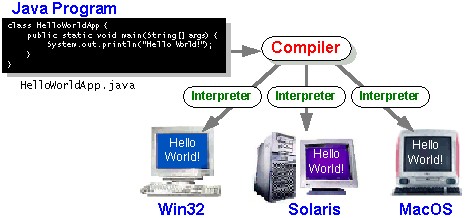


Fig.no 2.10.1 Working of Java Program

Java bytecode serves as the machine code instructions for the Java Virtual Machine (Java VM), enabling the "write once, run anywhere" paradigm. Every Java interpreter, whether it's a development tool or a web browser capable of running applets, functions as an implementation of the Java VM. By compiling a program into bytecode, it becomes platform-independent, allowing it to run on any system with a Java VM installed. This flexibility means that a Java program written on one platform, such as Windows 2000, can seamlessly execute on other platforms like a Solaris workstation or an iMac, as long as they support the Java VM.



**Fig.no 2.10.2 Implementation of Java Virtual Machine**

The Java Platform

A platform refers to the hardware or software environment where a program operates. Common platforms include Windows 2000, Linux, Solaris, and MacOS, which typically combine both operating system and hardware components. The Java platform, however, distinguishes itself by being a software-only platform that operates atop other hardware-based platforms. Comprising two main components, the Java platform consists of The Java Virtual Machine (Java VM): This serves as the foundation for the Java platform and is adapted to various hardware-based platforms.

he Java Application Programming Interface (Java API).

This encompasses a vast collection of pre-built software components offering various functionalities, such as graphical user interface (GUI) widgets. The Java API organizes these components into libraries of related classes and interfaces.

In the subsequent section, "What Can Java Technology Do?" highlights the functionalities provided by some of the packages within the Java API. The illustration below demonstrates a program operating within the Java platform, with the Java API and the virtual machine shielding the program from direct interaction with the hardware.

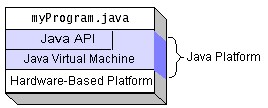


Fig.no 2.10.3 Program Running on the Java Platform

Native code refers to code that, once compiled, runs directly on a specific hardware platform. In contrast, the Java platform operates as a platform-independent environment, where programs can run across different hardware architectures. While the Java platform may exhibit slightly lower performance compared to native code, optimizations such as smart compilers, well-tuned interpreters, and just-in-time byte code compilers help bridge this performance gap while preserving portability.

**What Can Java Technology Do?**

Java technology offers a versatile set of programming options. Applets and applications are among the most common types of programs written in the Java. Applets, familiar to web users, adhere to specific conventions allowing them to run within Java-enabled browsers.

However, Java's capabilities extend beyond web applets. The Java programming language serves as a robust software platform.

Applications in Java can range from standalone programs to servers that support clients on a network. Servers, such as web servers, proxy servers, mail servers, and print servers, run directly on the Java platform, serving clients' needs.

The Java API supports a wide array of programs through packages of software components, providing diverse functionalities. A full implementation of the Java platform offers essential features like objects, strings, threads, numbers, input and output mechanisms, data structures, system properties, and date and time functionalities.

Furthermore, the Java API includes support for applets, networking (including URLs, TCP, UDP sockets, and IP addresses), internationalization for localized programs, comprehensive security measures (both low-level and high-level), software components known as JavaBeans, object serialization for lightweight persistence and communication via Remote Method Invocation (RMI), and Java Database

Connectivity (JDBC) for uniform access to relational databases.

Moreover, the Java platform provides APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The Java 2 SDK encompasses these functionalities, facilitating the development of diverse applications and enhancing the overall capabilities of the Java platform.

Installation of java

* Go to [http://www.oracle.com/technetwork/java/javase/downloads /index.html.](http://www.oracle.com/technetwork/java/javase/downloads%20/index.html)
* click on JDK DOWNLOAD button. run the exe file and then follow the instruction given in wizard.
* To set up the path
* Right click on my pc and then go to my properties

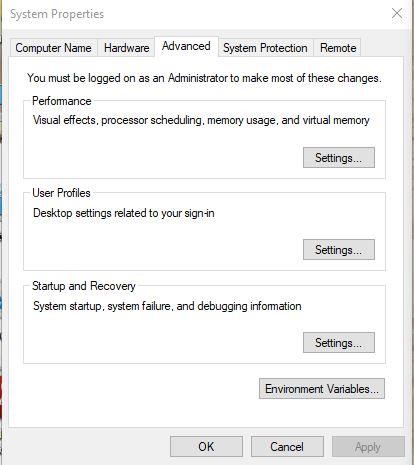
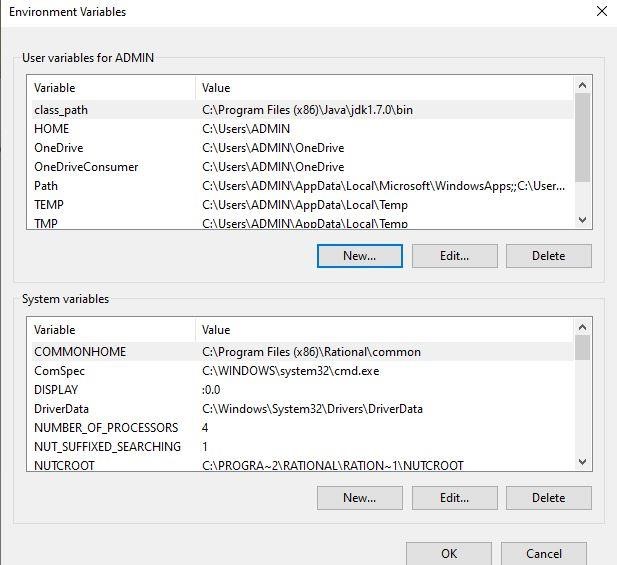


Fig.no 2.10.4 Properties wizard

* Go to advanced settings and then click on environment variables
* create a class path and copy the path of the java folder where it is located in program files.



**Fig.no 2.10.5 Path setting for java**

**Installation and setup of Apache Tomcat**

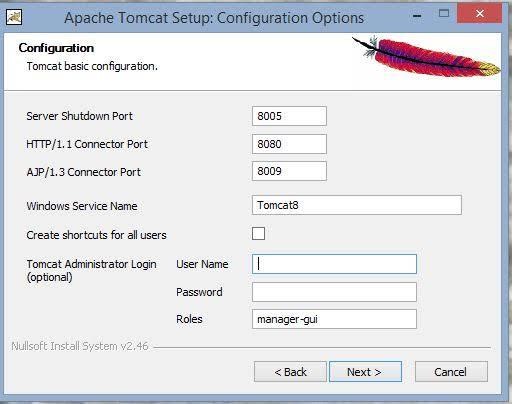
* Go to <http://tomcat.apache.org/index.html>and click on download latest versions.

* Run the exe file and click on next and follow the wizard instructions.



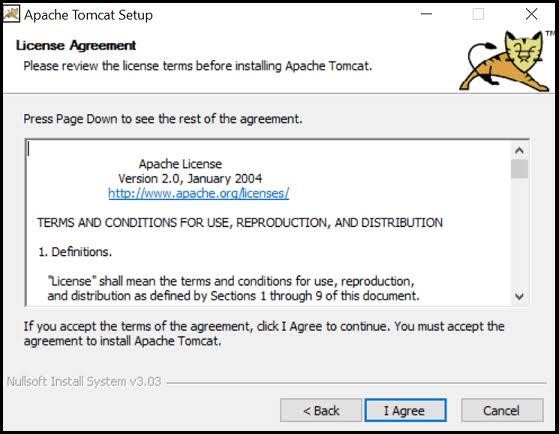
###### Fig.no 2.10.6 Welcome page of tomcat

* Click on install with port number 8090 with username and password as **aits** and  **aits.**
* Mention the connection port as 8090 and then click on next and finally click on finish.



###### Fig.no 2.10.7 Tomcat configuration options page

• Click on I agree button in license agreement in order to accept the terms and condition.



**Fig.no 2.10.8 Tomcat license agreement**

###### Installation and setup of SQL

* Go to<http://dev.mywql.com/downloads/>. and click on install button.
* After completion of installation, click on exe file and then click on next.Run the MySQL setup and click on next and follow the instruction in wizard.



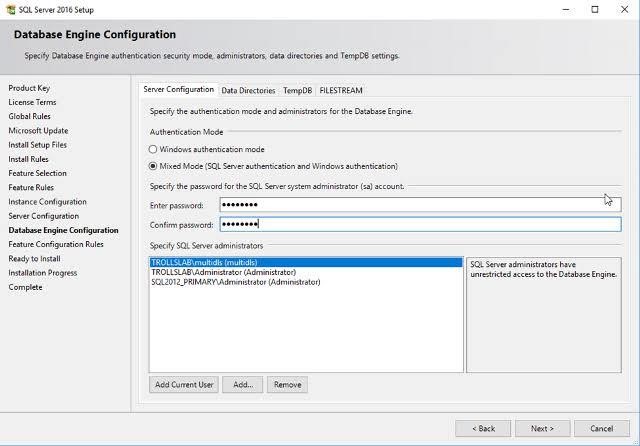
###### Fig.no 2.10.9 Welcome wizard of MySQL

• Conform the type as typical and then click on next and follow the instructions.



###### Fig.no 2.10.10 SQL setup wizard

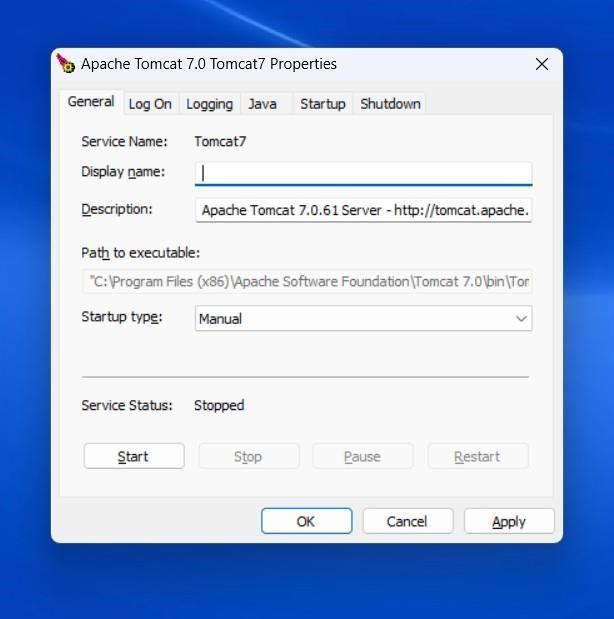
• Now confirm the password as root in system settings field and then click on finish.



###### Fig.no 2.10.11 Database configuration engine

• Tomcat start page serves as the entry point to the Tomcat web server, providing

Users with essential information and resources.



**Fig.no 2.10.12 Tomcat Page**

**SYSTEM DESIGN**

|  |  |
| --- | --- |
| System Design | Detecting Fraudulent Student Communication In A Multiple Choice Online Test Environment |

# 3. SYSTEM DESIGN

### 3.1 Database Design (ER-Diagram)

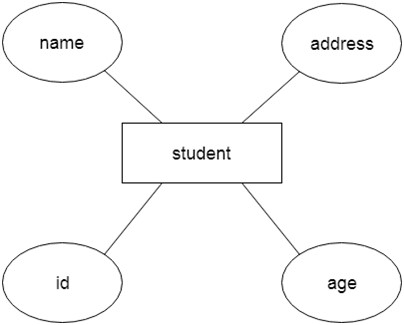
An Entity-Relationship (ER) model illustrates the structure of a database using a visual representation known as an Entity-Relationship Diagram (ER Diagram). This model serves as a blueprint for designing the database schema and capturing the relationships between different entities and attributes.

The ER model provides a systematic approach to organizing and conceptualizing the data within a database system. It represents entities as well as the relationships between them, helping to clarify how data elements are connected and organized.

##### 3.1.1 ER-Model

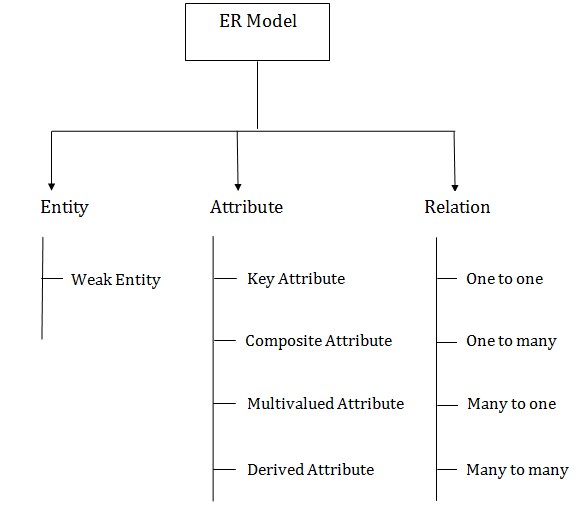
1. The Emergency Room model corresponds to an Entity-Relationship model serving as a high-level representation of data structures. It is utilized to illustrate the data components and relationships within a defined system.
2. It establishes a structured framework for the database. Moreover, it provides the straightforward and easily understandable perspective on the data.
3. In Entity-Relationship modeling,the organizational database structure is depicted depicted through a design known as an Entity-Relationship diagram.
4. For instance, consider designing a school database. An educational record could be represented as an entity with attributes such as name, ID, age, etc. Similarly, the address could be another entity with attributes like city, street name, zip code etc. and there would be a relationship between them.

26



**Fig.no 3.1.1 ER-Model**

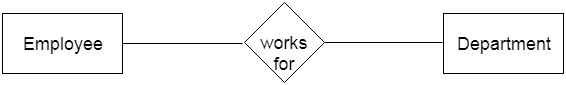
Component of ER Diagram



**Fig.no 3.1.2 Component of ER-Diagram**

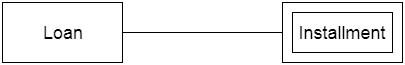
1. Entity

A substance may be anything, class, individual or spot. In the ER frame, a substance can be tended to as square shapes. Think about a relationship as a delineation chief, thing, specialist, office, etc can be taken as a substance.



1. Weak Entity

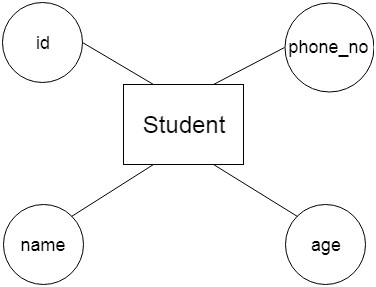
A substance that depends upon another component called a frail substance. The frail element contains no critical trait of its own. The feeble substance is addressed by a twofold square shape.



2. Characteristic

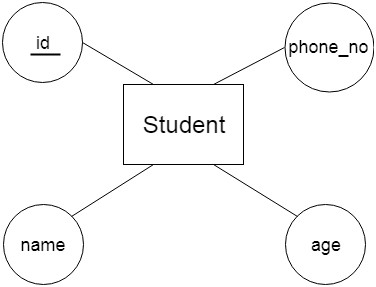
The quality is utilized to depict the property of a section. Obscure is utilized to address a quality.

For example, id, age, contact number, name, etc can be attributes of a student.



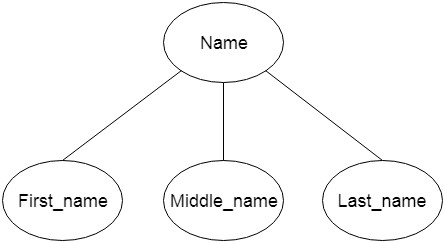
a. Key Attribute

The key quality is used to address the essential ascribes of a substance. It tends to a fundamental key. The key property is tended to by a circle with the text underlined**.**



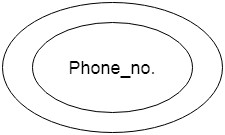
b. Composite Attribute

A property that made from various attributes is known as a composite quality. The composite trademark is tended to by an oval, and those circles are related with a circle.



c. Multivalued Attribute

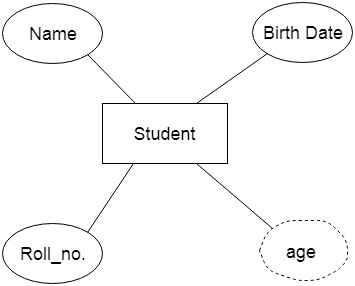
A quality can have more than one worth. These qualities are known as a multivalued property. The twofold oval is used to address multivalued property. For example, a student can have more than one phone num



c. Determined Attribute

A property that can be gotten from another quality is known as a decided attribute. It will in general be tended to by a ran circle.

For example, a singular's age changes long term and can be gotten from one more quality like Date of birth.



###### 3. Relationship

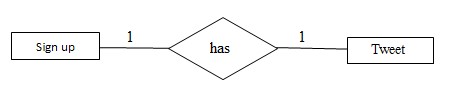
A relationship is used to depict the connection between substances. Important stone or rhombus is utilized to address the relationship.



**Sorts of relationship are as per the following**

a. One-to-One Relationship

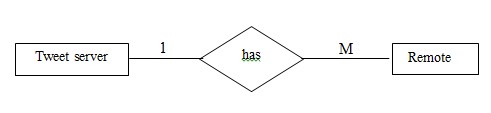
At the point when just a single instance of a component is connected with the relationship, then it is known as facilitated relationship. For instance, A female can wed to one male, and a male can wed to one female.



b. One-to-Many Relationship

Exactly when simply a solitary illustration of the substance on the left, and more than one event of a component on the right associates with the relationship then this is known as a one-to-various connections.

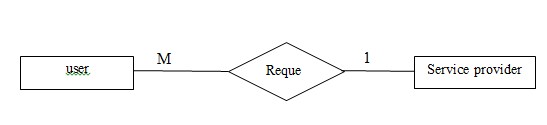
For example, Scientist can envision various manifestations, but the improvement is done by the really express analyst.



c. Many-to-One Relationship

Exactly when more than one event of the component on the left, and simply a solitary event of a substance on the right associates with the relationship then it is known as a many-to-one relationship.

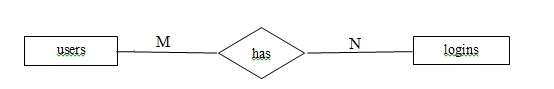
For example, Student enrolls for only a solitary course, but a course can have various students.



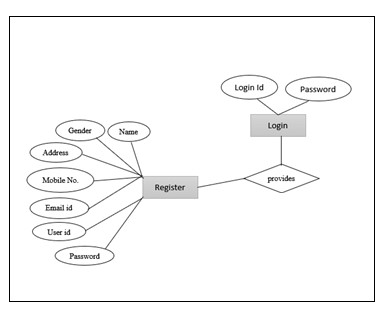
d. Many-to-Many Relationship

At the point when more than one event of the substance on the left, and more than one event of a component on the right associate with the relationship then it is known as a many-to-various connections.

For example, Employee can allot by numerous exercises and project can have various specialists.



E-R Diagram



**Fig.no 3.1.3 E-R Diagram**

### 3.2. Data Dictionary

A Data Dictionary compiles names, definitions, and attributes concerning data elements utilized or stored within a database, information system, or part of a research project. It delineates the meanings and functions of data elements within the context of a project and offers guidance on understanding, recognizing meanings, and description.

Additionally, a Data Dictionary offers metadata about data elements, aiding in defining the scope and attributes of data elements, as well as the guidelines for their usage and application.

Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Collation | Null | Default |
| Id | Int (11) |  | No | None |
| User name | Varchar (30) | Latin\_swedish\_ci | No | None |
| email | Varchar (30) | Latin\_swedish\_ci | No |  |
| pswd | Varchar (30) | Latin\_swedish\_ci | No |  |
| phone no | Varchar (30) | Latin\_swedish\_ci | No |  |
| country | Varchar (30) | Latin\_swedish\_ci | No |  |
| state | Varchar (30) | Latin\_swedish\_ci | No |  |

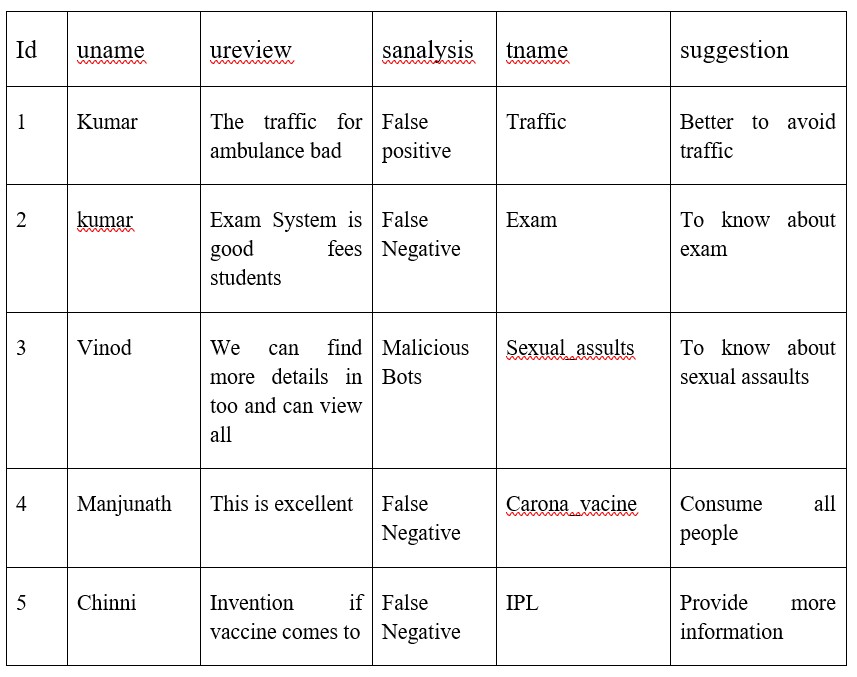
**Table.no 3.2.1 Data Dictionary**

Data Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Id | User name | email | Pswd | phno | country | state | city |
| 1 | vijay | vkappati@ gmail.com | \*\*\*\*\*\*\*\* | 6281407028 | India | AP | kurnool |
| 2 | vinod | vinod@gm  ail.com | \*\*\*\*\*\*\*\* | 8790557392 | India | AP | kurnool |
| 3 | Ajay | ajay@gmai l.com | \*\*\*\*\*\*\*\* | 7780490892 | India | AP | mdk |
| 4 | vishnu | Vishnukvr @gmail.co m | \*\*\*\*\*\*\*\* | 9849466191 | India | AP | rzp |

**Table.no 3.2.2 Data Table**

Received User Data Analysis



**Table.no 3.2.3 Received User Data Analysis**

### 3.3. Normalization

Normalization is the primary method for optimizing data in a database to fulfill two essential criteria. Data dependencies are logical, ensuring that all related data items are stored together.

Normalization is crucial for various reasons, primarily because it enables databases to occupy minimal disk space, resulting in enhanced performance.

Normalization is also referred to as data standardization.The three primary types of normalization are outlined below. Note: "NF" stands for "normal form."

First typical structure (1NF)

Tables in 1NF should comply with certain standards:

1. Every cell should contain just a solitary (nuclear) esteem.
2. Each part in the table ought to be astoundingly named.
3. All characteristics in a part ought to connect with a comparative region.

1NF

|  |  |  |
| --- | --- | --- |
| **UserID** | **Username** | **Password** |
| 015 | John | \*\*\*\*\*\*\*\* |
| 016 | Princess | \*\*\*\*\*\*\*\* |
| 027 | Tom | \*\*\*\*\*\*\*\* |
| 028 | Claire | \*\*\*\*\*\*\*\* |
| 029 | Robert | \*\*\*\*\*\*\*\* |

**Table.no 3.3.1 1NF**

Second typical structure (2NF)

Tables in 2NF ought to be in 1NF and not have any most of the way dependence

(e.g., each non-prime quality ought to be dependent upon the table's fundamental key)

2NF

|  |  |  |  |
| --- | --- | --- | --- |
| **User Id** | **Received Data through IOT** | **pswd** | **Login** |
| 1 | 11 | \*\*\*\*\*\*\*\* | Signup |
| 2 | 12 | \*\*\*\*\*\*\*\* | Signup |
| 3 | 13 | \*\*\*\*\*\*\*\* | Signup |
| 4 | 14 | \*\*\*\*\*\*\*\* | Signup |
| 5 | 15 | \*\*\*\*\*\*\*\* | Signup |

**Table.no 3.3.2 2NF**

Third ordinary structure (3NF)

Tables in 3NF ought to be in 2NF and have no transitive reasonable circumstances on the fundamental key. The going with two NFs furthermore exists anyway are only here and there used.

USER DETAILS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **NAME** | **EMAIL** | **STATE** | **CITY** | **COUNTRY** |
| 11 | Vishnu | vishnu@gmail.com | AP | RZP | INDIA |
| 12 | Vijay | vijay@gmail.com | AP | RZP | INDIA |
| 13 | Vinod | vinod@gmail.com | AP | RZP | INDIA |
| 14 | Ramu | Ramu@gmail.com | AP | RZP | INDIA |

**Table.no 3.3.3 User Details**

USER DETAILS

|  |  |  |
| --- | --- | --- |
| **USER ID** | **PASSSWORD** | **LOGIN** |
| Server | \*\*\*\*\*\*\* | Signup |
| Vijay | \*\*\*\*\*\*\* | Signup |

**Table.no 3.3.4 User details**

Boyce-Codd Normal Form (BCNF)

Normalization is a critical process in database management aimed at organizing tables to minimize anomalies and ensure data integrity. It follows a series of stages known as normal forms. These normal forms help structure tables efficiently and reduce redundancy and inconsistency in data.

**Unnormalized Form (UNF):** The initial state of a table where data is not organized according to any specific rules.

**First Normal Form (1NF):** In 1NF, each column contains atomic values, and there are no repeating groups or arrays within a row.

**Second Normal Form (2NF):** 2NF requires that every non-key attribute be fully functionally dependent on the primary key.

**Third Normal Form (3NF):** In 3NF, no transitive dependencies should exist, meaning that non-key attributes should not depend on other non-key attributes.

**Elementary Key Normal Form (EKNF):** EKNF is a further refinement of 3NF, emphasizing the use of elementary keys.

Boyce-Codd Normal Form (BCNF): BCNF addresses anomalies that may arise when multiple candidate keys exist. It requires that for every non-trivial functional dependency (X → Y), X must be a super key.

Fourth Normal Form (4NF): To achieve 4NF, a table must be in BCNF and should not have multi-valued dependencies.

Essential Tuple Normal Form (ETNF): ETNF is a condition where each attribute in a tuple is essential to the understanding of the tuple itself.

Normal forms

|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | An alliance is in 1NF enduring it contains an atomic worth. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | An association will be in 2NF expecting it is in 1NF and all nonkey credits are totally down to earth ward on the fundamental key. |
| [3NF](https://www.javatpoint.com/dbms-third-normal-form) | An alliance will be in 3NF enduring it is in 2NF and no change dependence exists. |
| BCNF | A more grounded importance of 3NF is known as Boyce Codd's common design. |
| [4NF](https://www.javatpoint.com/dbms-forth-normal-form) | An association will be in 4NF expecting it is in Boyce Codd's common place construction and has no multi-regarded dependence. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | An association is in 5NF. In case it is in 4NF and contains no join dependence, joining should be lossless. |

**Table.no 3.3.5 Normal forms**

**Benefits of Normalization**

Reduction of data redundancy: Normalization helps eliminate redundant data by organizing it efficiently across tables. Improved overall database organization by structuring data according to normalization rules, databases become more organized and easier to manage. Data consistency within the database.Normalization ensures that data remains consistent across tables, reducing the risk of inconsistencies. More flexible database design.

Normalization allows for more flexibility in database design, making it easier to accommodate changes and updates. Upholds the principle of data integrity.

Normalization promotes data integrity by minimizing anomalies and ensuring accurate representation of data relationships.

**Disadvantages of Normalization**

Careless decomposition: If normalization is done without a clear understanding of user requirements, it can lead to excessive decomposition and unnecessary complexity in the database design.

Decreased performance: As tables are normalized to higher normal forms such as 4NF, 5NF, it may lead to decreased performance due to increased join operations and complexity in querying the database.

### 3.4. UML Diagrams

UML is a standardized general-purpose modelling language in the field of objectoriented software engineering. UML includes a set of graphic notation techniques to create visual models of object-oriented software systems. UML combines techniques from data modelling, business modelling, object modelling, and component modelling and can be used throughout the software development life-cycle and across different implementation technologies.

UML is a standardized general-purpose modelling language used for Visualization, Documentation and construction of the system. Modelling A model is a simplification at some level of abstraction We build models to better understand the systems we are developing. There is a difference between a UML model and the set of diagrams of a system.

UML? Complete History The 1990s was the era of development of object-oriented languages such as C++. These object- oriented languages were used to create complex but compelling systems. As the systems developed were complicated to understand, it led to the design and analysis problems which were faced after the deployment of the system. It was difficult to explain the system to others.

As soon as the UML was introduced, many game-changing experiments and approaches were made for simplifying such difficult tasks of analyzing the system. UML is an object-oriented unified modeling language. It was invented by brilliant software engineers Grady Booch, Ivar Jacobson, and James Rumbaugh of Rational software during 1994 and 1995. It was under development until 1996. Each of UML inventors, viz, Grady Booch, Ivar Jacobson, and James Rumbaugh had a fantastic idea for designing a language which will reduce the complexity

* Booch’s method was very flexible to work with during the design and construction of objects.
* Jacobson’s method provided a great way to work around use-cases. It also has a  powerful approach for high-level design.
* Rumbaugh’s method turned out to be very useful while handling sensitivesystems.

An Overview of UML

**T**he Unified Modeling Language is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system.

The UML is appropriate for modeling systems ranging from enterprise information systems to distributed Web-based applications and even to hard real time embedded systems. The UML is a language for

* Visualizing
* Specifying
* Constructing
* Documenting
* Visualizing

UML Building Blocks

The vocabulary of the UML encompasses three kinds of building blocks

1. Things
2. Relationships
3. Diagrams

Things are the abstractions that are first-class citizens in a model relationship tie these things together; diagrams group interesting collections of things.

Things

There are four kinds of things in the UML

1. Structural things
2. Behavioral things
3. Grouping things
4. Annotational things

Structural Things

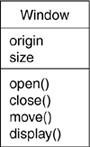


Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. Collectively, the structural things are called classifiers.

Class

A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically, a class is rendered as a rectangle, usually including its name, attributes.

Notation



Sample Class Diagram

Interface

An interface is a collection of operations that specify a service of a class or component. An interface therefore describes the externally visible behavior of that element. An interface might represent the complete behavior of a class or component or only a part of that behavior. An interface defines a set of operation specifications (that is, their signatures) but never a set of operation implementations.

Notation

Sample Interface Diagram

Collaboration

A collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that bigger than the sum of all the elements. Collaborations have structural, as well as behavioral. A given class might participate in several collaborations.Graphically, a collaboration is rendered as an ellipse with dashed lines, sometimes including only its name.

Notation

**Collaboration\_Name**

Sample Collaboration Diagram

Use Case

A use case is a description of sequences of actions that a system performs that yield observable results of value to a particular actor. A use case is used to structure the behavioral things in a model. A use case is realized by a collaboration. Graphically, a use case is rendered as an ellipse with solid lines, usually including only its name.

Notation

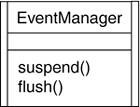


Sample use case diagram

Active Class

An active class is just like a class except that its objects represent elements whose behavior is concurrent with other elements. Graphically, an active class is rendered as a class with double lines on the left and right; it usually includes its name, attributes, and operations.

Notation



Sample Active Class Diagram

Component

A component is a modular part of the system design that hides its implementation behind a set of external interfaces. Within a system, components sharing the same interfaces can be substituted while preserving the same logical behavior. The implementation of a component can be expressed by wiring together parts and connectors; the parts can include smaller components. Graphically, a component is rendered like a class with a special icon in the upper right corner.

Notation



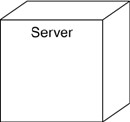
Sample Component Diagram

###### Node

A node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and, often, processing capability. A set of components may reside on a node and may also migrate from node to node.

Graphically, a node is rendered as a cube, usually including only its name.

Notation



Sample Node Diagram

Behavioral Things

Behavioral things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space.

Interaction

An interaction is a behavior that comprises a set of messages exchanged among a set of objects or roles within a particular context to accomplish a specific purpose. The behavior of a society of objects or of an individual operation may be specified with an interaction. An interaction involves a number of other elements, including messages, actions, and connectors (the connection between objects). Graphically, a message is rendered as a directed line, almost always including the name of its operation.

Notation

display

Sample Interaction Diagram

State Machine

A state machine is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its responses to those events. The behavior of an individual class or a collaboration of classes may be specified with a state machine. A state machine involves a number of other elements, including states, transitions (the flow from state to state), events (things that trigger a transition), and activities (the response to a transition).

Graphically, a state is rendered as a rounded rectangle, usually including its name.

Notation



Sample State Machine

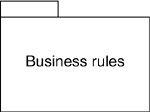
Grouping Things

Grouping things are the organizational parts of UML models. These are the boxes into which a model can be decomposed. There is one primary kind of grouping thing, namely, packages.

Packages

A package is a general-purpose mechanism for organizing the design itself, as opposed to classes, which organize implementation constructs. Structural things, behavioral things, and even other grouping things may be placed in a package. Unlike components (which exist at run time), a package is purely conceptual (meaning that it exists only at development time). Graphically, a package is rendered as a tabbed folder, usually including only its name and, sometimes, its contents.

Notation



Sample Package Diagram

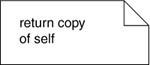
Annotation Things

Annotational things are the explanatory parts of UML models. These are the comments you may apply to describe, illuminate, and remark about any element in a model. There is one primary kind of annotational thing, called a note.

Note

A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements. Graphically, a note is rendered as a rectangle with a dog-eared corner, together with a textual or graphical comment.

Notation



Sample Note Diagram

UML Relationships

There are four kinds of relationships in the UML

1. Dependency
2. Association
3. Generalization
4. Realization

Dependency

A dependency is a semantic relationship between two model elements in which a change to one element (the independent one) may affect the semantics of the other element (the dependent one). Graphically, a dependency is rendered as a dashed line, possibly directed, and occasionally including a label.

Notation



Dependency

Association

A association is a structural relationship among classes that describes a set of links, a link being a connection among objects that are instances of the classes. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts. Graphically, an association is rendered as a solid line, possibly directed, occasionally including a label, and often containing other adornments.

Notation



Association

Generalization

A generalization is a specialization generalization relationship in which the specialized element (the child) builds on the specification of the generalized element (the parent). The child shares the structure and the behavior of the parent. Graphically, a generalization relationship is rendered as a solid line with a hollow arrowhead pointing to the parent.

Notation



Generalization

Realization

A realization is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out. You'll encounter realization relationships in two places: between interfaces and the classes or components that realize them, and between use cases and the collaborations that realize them. Graphically, a realization relationship is rendered as a cross between a generalization and a dependency relationship.

Notation

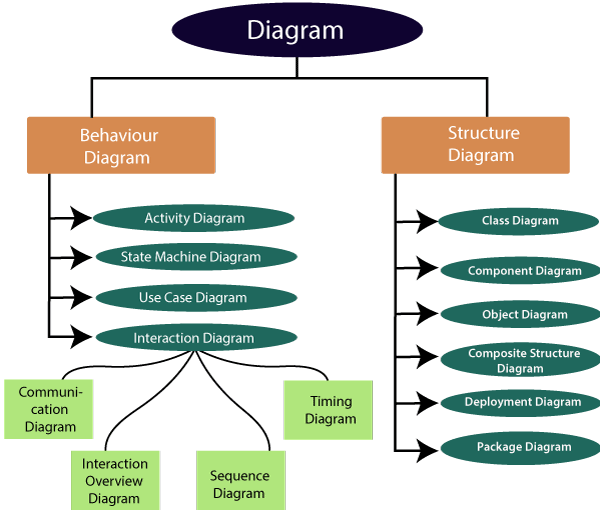


Realization

UML Diagrams

Diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships). In theory, a diagram may contain any combination of things and relationships. For this reason, the UML includes nine such diagrams.

* Class Diagram
* Use case diagram
* Sequence diagram
* Collaboration diagram
* State chart diagram
* Activity diagram
* Component diagram
* Deployment diagram



**Fig.no 3.4.1 UML Diagram**

USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram described and expressed using a Use-case analysis. Its objective is to present a graphical representation of the value provided by a system in terms of actors, their goals (represented as use cases), and any relationships between those use cases. The primary purpose of a use case diagram is to illustrate which system functions are performed for which actor.

Authorize Users

Create Masters

Admin

Logins

Register

Write Exam

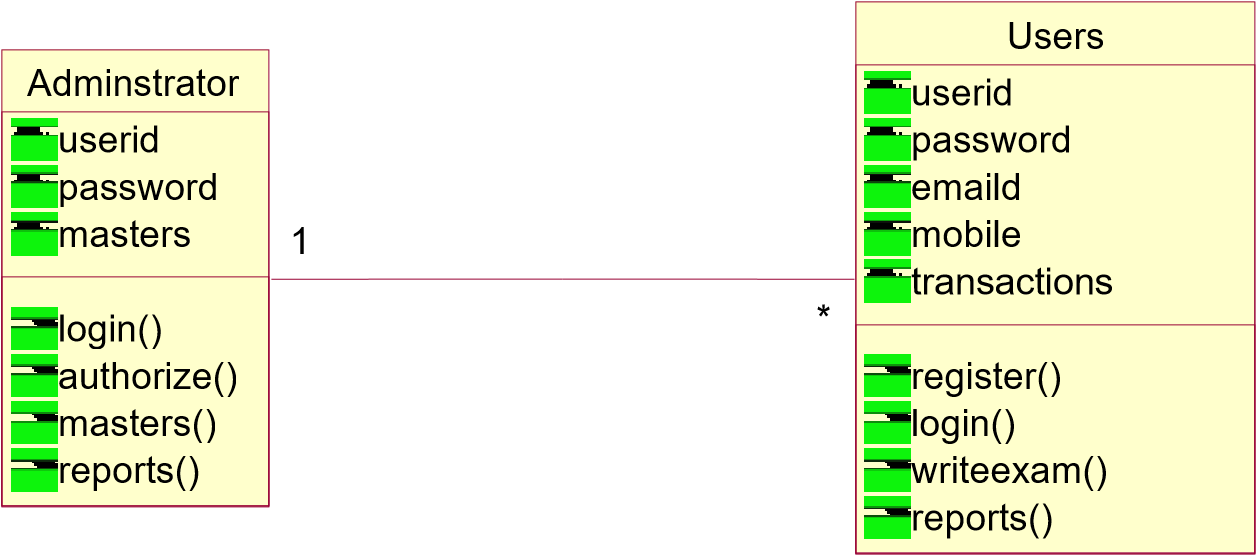
Usesr

Reports

**Fig.no 3.4.2 Use Case Diagram**

CLASS DIAGRAM

In computer programming, a class diagram in the Unified Modeling Language (UML) is a type of static structural diagram that illustrates the architecture of a system by displaying the structure's classes, their properties, operations (or methods), and the relationships among the classes. It delineates which class holds data.



**Fig.no 3.4.3 Class Diagram**

SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a type of communication diagram that illustrates how processes interact with each other and the order in which they occur. It is a variation of a Message Sequence Chart. Sequence diagrams are sometimes referred to as event diagrams, event scenarios, or timing diagrams.

**Fig.no 3.4.4 Sequence Diagram**

User

Admin

Server

login

1:

2:

validate

4:

Create Masters

3:

login

to

admin menu

Authorize Users

6:

5:

Register

login

7:

8:

validate

10:

write exam

11:

Reports

9:

Reports

ACTIVITY DIAGRAM

It models the flow of control from one activity to the other. With the help of an activity diagram, we can model sequential and concurrent activities. It visually depicts the workflow as well as what causes an event to occur.

Get Details

Enter User

Id and

V

Rejected

V

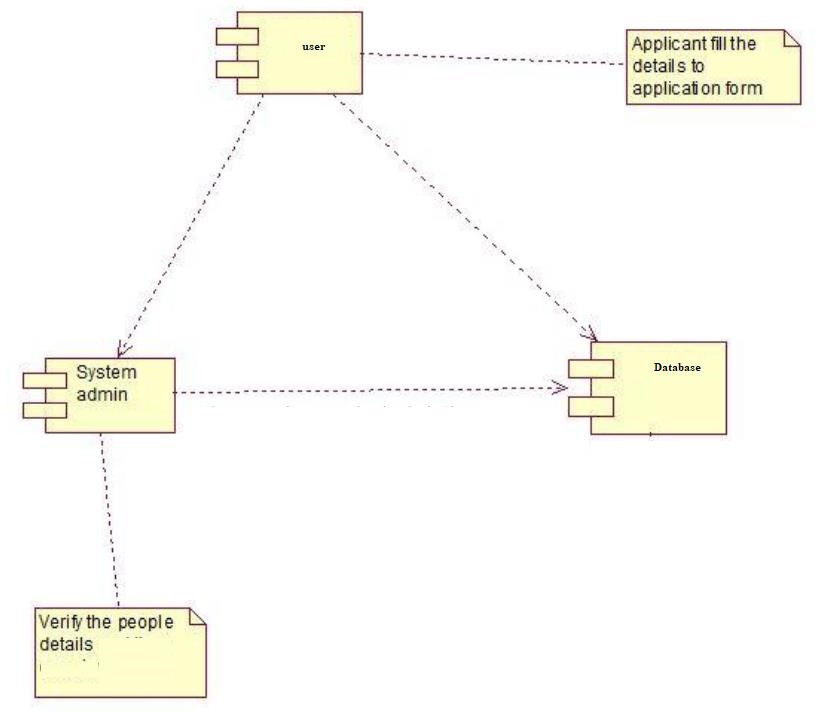
Invalid User

**Fig.no 3.4.5 Activity Diagram**

COMPONENT DIAGRAM

A component diagram is used to break down a large object-oriented system into smaller components, making them more manageable. It provides a physical view of a system, such as executables, files, libraries, etc., that reside within the node.

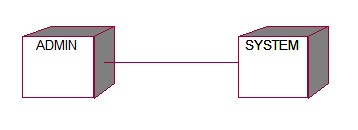
It visualizes not only the relationships but also the organization between the components present in the system. It assists in forming an executable system. A component is a single unit of the system, which is replaceable and executable.



**Fig.no 3.4.6 Component Diagram**

DEPLOYEMENT DIAGRAM

The deployment diagram illustrates the physical hardware on which the software will be deployed. It represents the static deployment view of a system, depicting the nodes and their relationships. It details how software is distributed across the hardware. The deployment diagram maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. As it involves multiple nodes, the relationships are depicted using communication pathways.



**Fig.no 3.4.7 Deployment Diagram**

COLLABORATION DIAGRAM

In a collaboration diagram, the sequence of method calls is depicted using a numbering technique to illustrate how methods are called sequentially. This numbering indicates the order in which the methods are invoked. Let's consider the order management system to describe the collaboration diagram. The method calls in a collaboration diagram resemble those in a sequence diagram. However, the key difference lies in the fact that while the sequence diagram focuses solely on method invocation, the collaboration diagram provides additional information by illustrating the organization of objects involved in the interactions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User | 2:  validat  e  5:  Register  7:  login  8:  validat  e  Perform Transacti  Reports | |  | |
| 10: 11: | ~~1~~  :  login  3:  login to admin menu  4:  Create Masters  6:  Authorize Users  9:  Repo  ~~rt~~  s  ons | Admin |
|  |
|  | Server |

**Fig.no 3.4.8 Collaboration Diagram**

**TESTING**

|  |  |
| --- | --- |
| Testing | Detecting Fraudulent Student Communication In A Multiple Choice Online Test Environment |

# 4. TESTING

## 4.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

Testing is one of the most important phases in the software development activity. In software development life cycle (SDLC), the main aim of testing process is the quality; the developed software is tested against attaining the required functionality and performance.

During the testing process the software is worked with some particular test cases and the output of the test cases are analyzed whether the software is working according to the expectations or not.

Levels of Testing Since the errors in the software can be injured at any stage. So, we have to carry out the testing process at different levels during the development. The basic levels of testing are Unit, Integration, System and Acceptance Testing.

Detecting Fraudulent Student Communication In

A Multiple Choice Online Test Environment

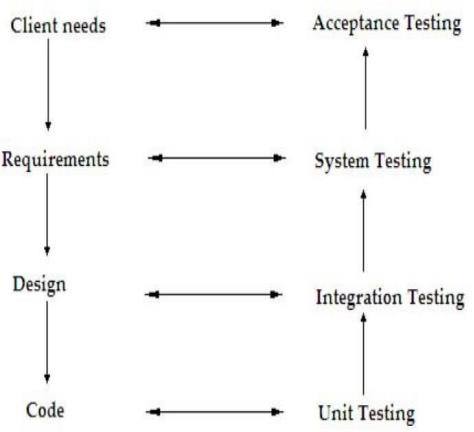


Fig.no 4.1 Levels of Testing

The Unit Testing is carried out on coding. Here different modules are tested against the specifications produced during design for the modules. In case of integration testing different tested modules are combined into sub systems and tested in case of the system testing the full software is tested and in the next level of testing the system is tested with user requirement.

58

Detecting Fraudulent Student Communication In

A Multiple Choice Online Test Environment

## 4.2. TESTING METHODOLOGIES

The following are the Testing Methodologies

* Unit Testing
* Integration Testing
* User Acceptance Testing
* Output Testing
* Approval Checking

#### 4.2.1 Unit Testing

Unit testing puts together check exertion with respect to the smallest unit of Software plan that is the module. Unit testing. rehearses unequivocal courses in a module's control development to ensure all out consideration and most outrageous screw up distinguishing proof. This test revolves around each module independently, ensuring that it limits suitably as a unit. Subsequently, the naming is Unit Testing. During this testing, each module is attempted solely and the module points of communication are Affirmed for the consistency with plan assurance. Extremely huge taking care of way are pursued for the ordinary results. All bumble dealing with ways are moreover attempted.

Coordination Testing

Coordination testing settle the issues related with the twofold issues of affirmation and program improvement. After the item has been incorporated a bunch of high request tests are directed. The primary goal of testing process is to take unit attempted modules and collects a program structure that has been coordinated by plan.

#### 4.2.2 Integration Testing

Top-Down Integration

This method is a languid procedure for managing the progress of program structure. Modules are composed by moving lower through the control request, beginning with the chief program module. The module subordinates to the essential program module are coordinated into the development in either a significance first or breadth first way. The product is tried from primary module and individual stubs are supplanted when the test continues downwards.

Base up Integration

This system begins the new turn of events and testing with the modules basically level in the program structure. Since the modules are facilitated from the base up, taking care of expected for modules subordinate to a given level is by and large available and the prerequisite for nails is shed. The base up fuse system may be executed with the going with progresses.

1. The low-level modules are joined into bundles into bunches that playout aspecific Software sub-work.
2. A driver (i.e.) the control program for testing is made to work with information and result.
3. The bunch is tried.
4. Drivers are disposed of and bunches are joined moving vertical in the program structure.
5. The granular perspectives test every module independently and afterward every module will be module is incorporated with a primary module and tried for use fulness.

#### 4.2.3 Client Acceptance Testing

Client Acceptance of a structure is the indispensable part for the advancement of any system. The structure practical is pursued for client affirmation by constantly keeping in touch with the arranged system clients at the hour of making and making changes any spot required. The system made gives a well-disposed UI that can undoubtedly be another seen even by an individual to the framework.

#### 4.2.4 Output Testing

In the wake of playing out the endorsement testing, the ensuing stage is yield attempting of the proposed system, since no structure could be useful if it doesn't convey the essential outcome in the predefined plan. Getting some data about the setup expected by them tests the outcomes made or displayed by the structure feasible. Consequently, the outcome configuration is considered in 2 ways - one is on screen and one more in printed plan.

#### 4.2.5 Approval Checking

Approval checks are performed on the accompanying fields.

### 4.3 SOFTWARE TESTING STRATEGY

A system for structure testing organizes structure examinations and plan methodologies into an overall organized series of steps that results in the productive advancement of programming. The testing procedure must collaborate test organizing, try setup, test execution, and the resultant data grouping and appraisal. A strategy for programming testing ought to oblige low-level tests that are critical to affirm that a little source code part has been precisely executed as well as irrefutable level tests that endorse huge system limits against client requirements.

Programming testing is a fundamental part of programming quality affirmation and addresses an authoritative study of specific arrangement and coding. Testing tends tointriguing irregularity for the item. Accordingly, a movement of testing is performed for the proposed structure before the system is ready for client affirmation testing.

#### 4.3.1 FRAMEWORK TESTING

Programming once supported ought to be gotten together with other structure parts (for instance Equipment, people, informational collection). Structure testing affirms that all of the parts are real and that overall system work execution is achieved. It moreover tests to find irregularities between the structure and its novel objective, current subtleties and system documentation.

UNIT TESTING

This philosophy begins the new turn of events and testing with the modules basically level in the program structure. Unit testing is central for affirmation of the code conveyed during the coding stage, and hence the goals to test within reasoning of the modules. Including the distinct arrangement portrayal as an assistant, critical Conrail ways are attempted to reveal bungles inside the constraint of the modules.

This testing is finished during the programming stage itself. Sort of testing step, every module was viewed as working agreeably as respects to the normal result from the module. In Due Course, most recent innovation headways will be thought about. As a feature of specialized develop a large number of the systems administration framework will be nonexclusive in nature so future undertakings can either utilize or cooperate with this. What the future holds a ton to propose to the new development and refinement of this endeavor.

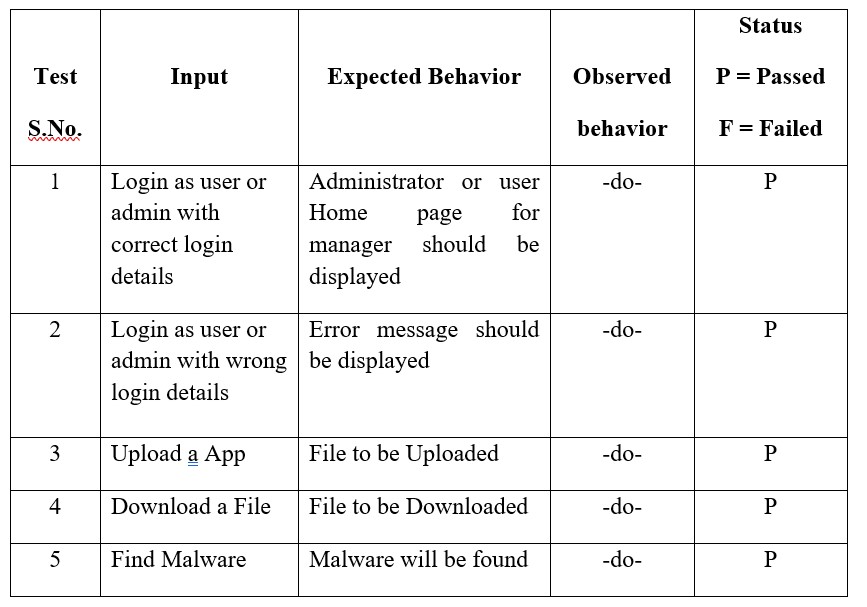
In system testing, coordination testing passed parts are taken as data. The goal of blend testing is to recognize any irregularity between the units that are integrated together. System testing recognizes gives up inside both the planned units and the whole structure. The delayed consequence of system testing is the seen approach to acting of a section or a structure when it is attempted.

### 4.4. Test Cases

An experiment is a report, which has a lot of test data, preconditions, expected results and postconditions, made for a particular test circumstance to really take a look at consistence against a specific need.

Try goes probably as the early phase for the test execution, and ensuing to applying a lot of information regards, the application has a legitimate outcome and leaves the structure at some end point or generally called execution postcondition.

Test cases



**Table.no 4.4 Test cases**

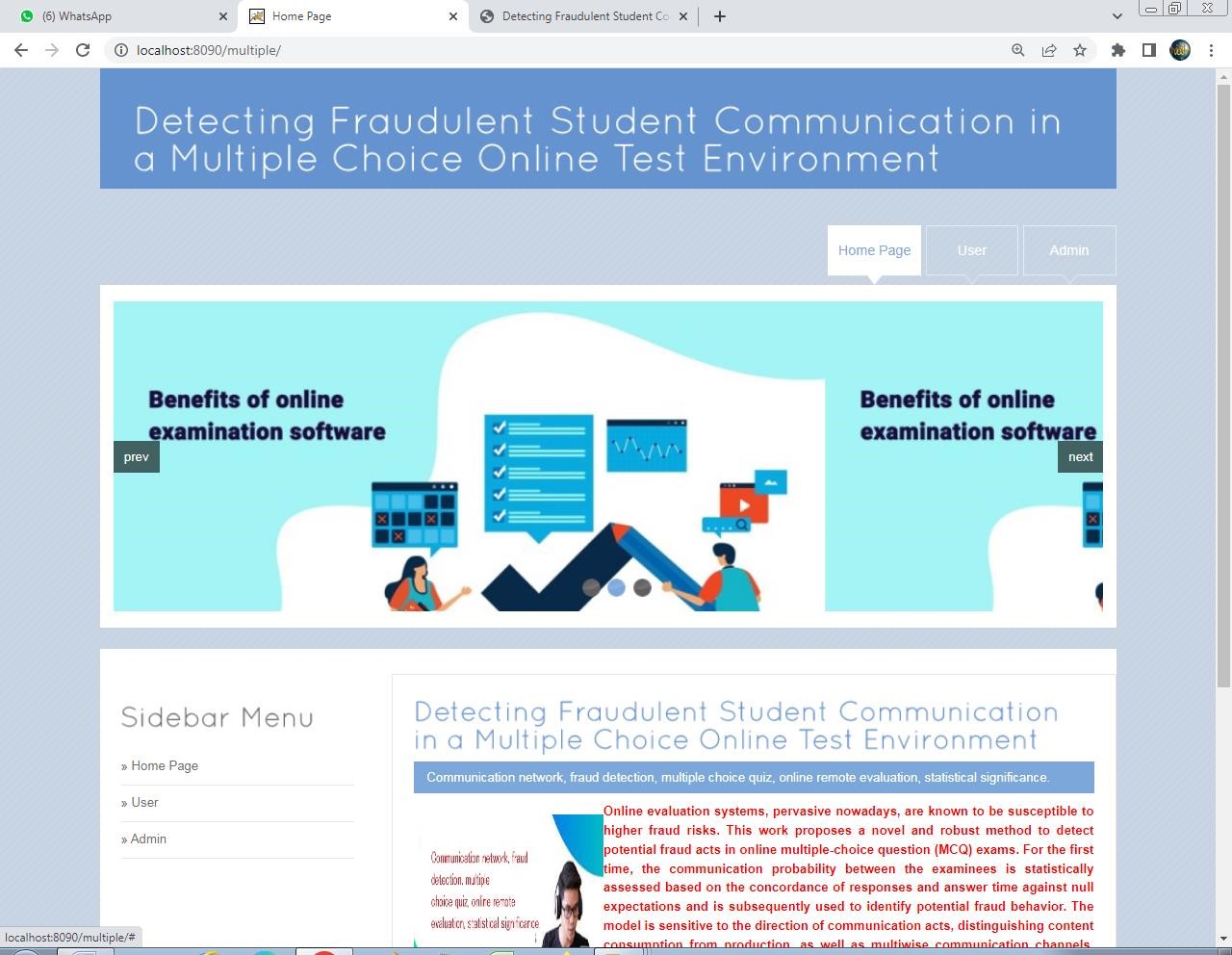
**IMPLEMENTATION**

|  |  |
| --- | --- |
| Implementation | Detecting Fraudulent Student Communication In A Multiple Choice Online Test Environment |

# 5. IMPLEMENTATION

### 5.1. Sample Screens

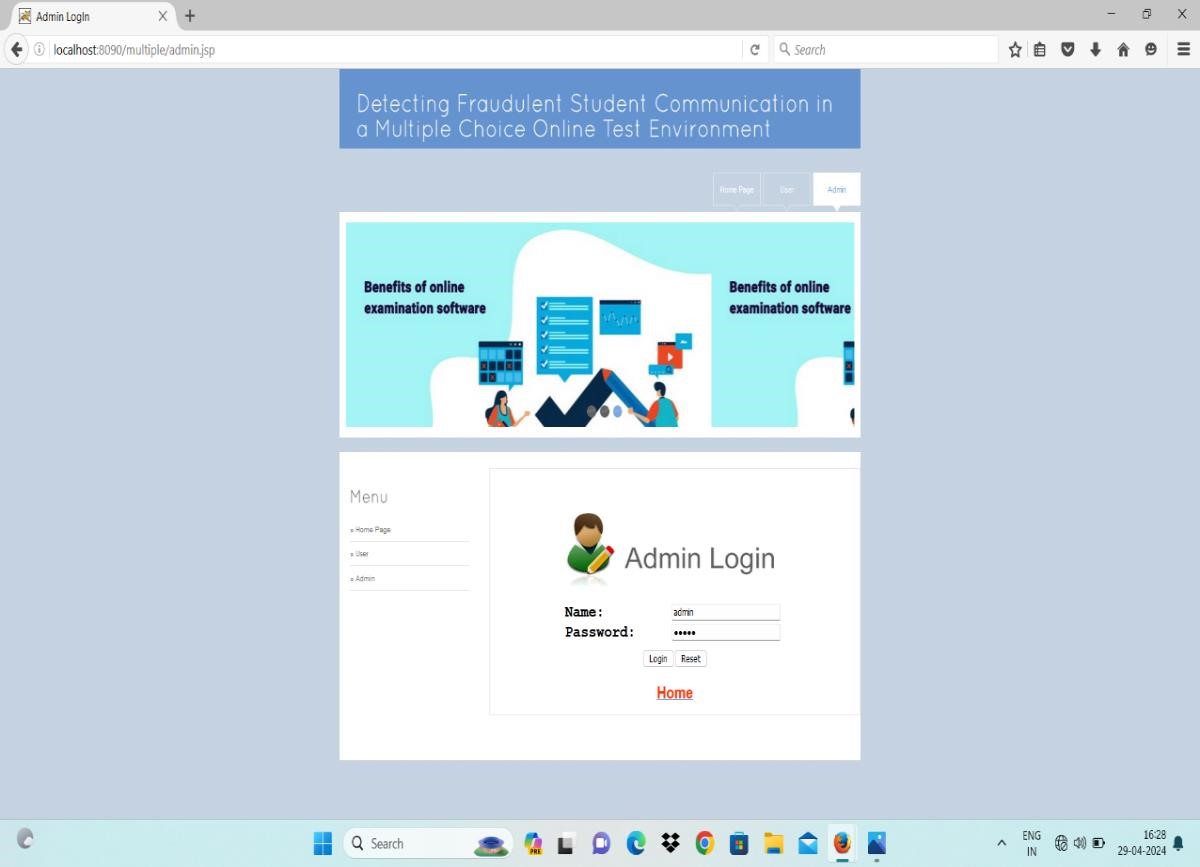
Screen 5.1.1: Home Page



Screen 5.1.1: Home Page

**Description:** This page about the Home page.

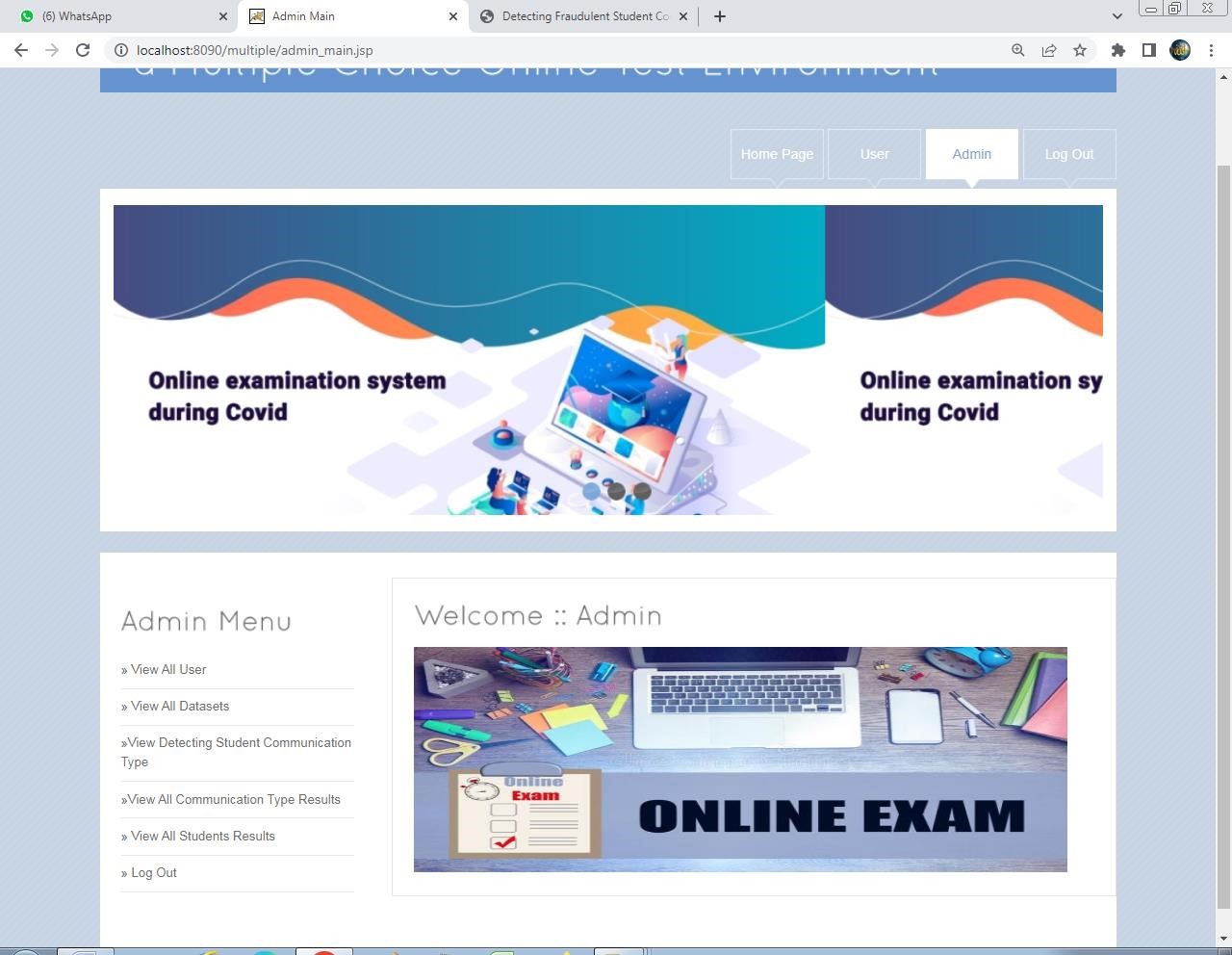
**Screen 5.1.2:** Admin Login page



Screen 5.1.2: Admin Login page

**Description:** This page about the admin login page

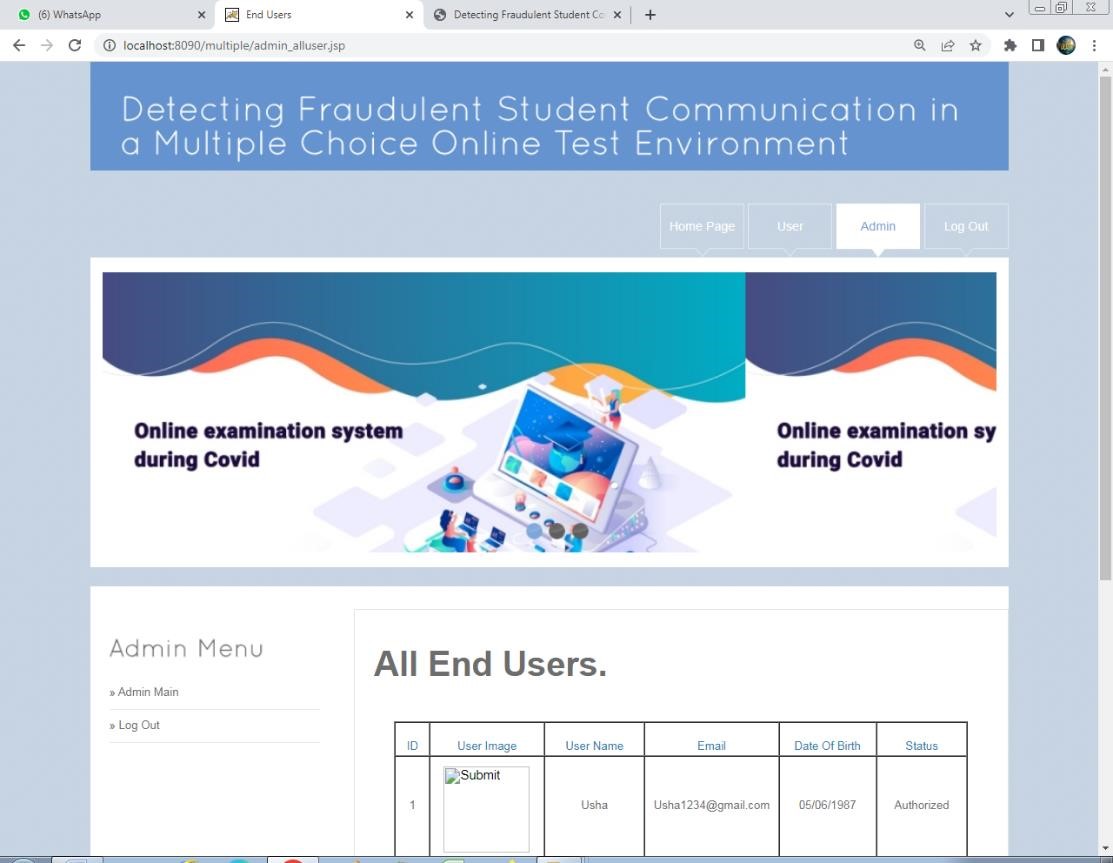
**Screen 5.1.3:** Admin Menu Page



Screen 5.1.3: Admin Menu Page

**Description**: In the above screen shows the admin menu page

**Screen 5.1.4:** View and Authorize



Screen 5.1.4: View and Authorize

**Description:** In the above screen admin can view the users and authorize

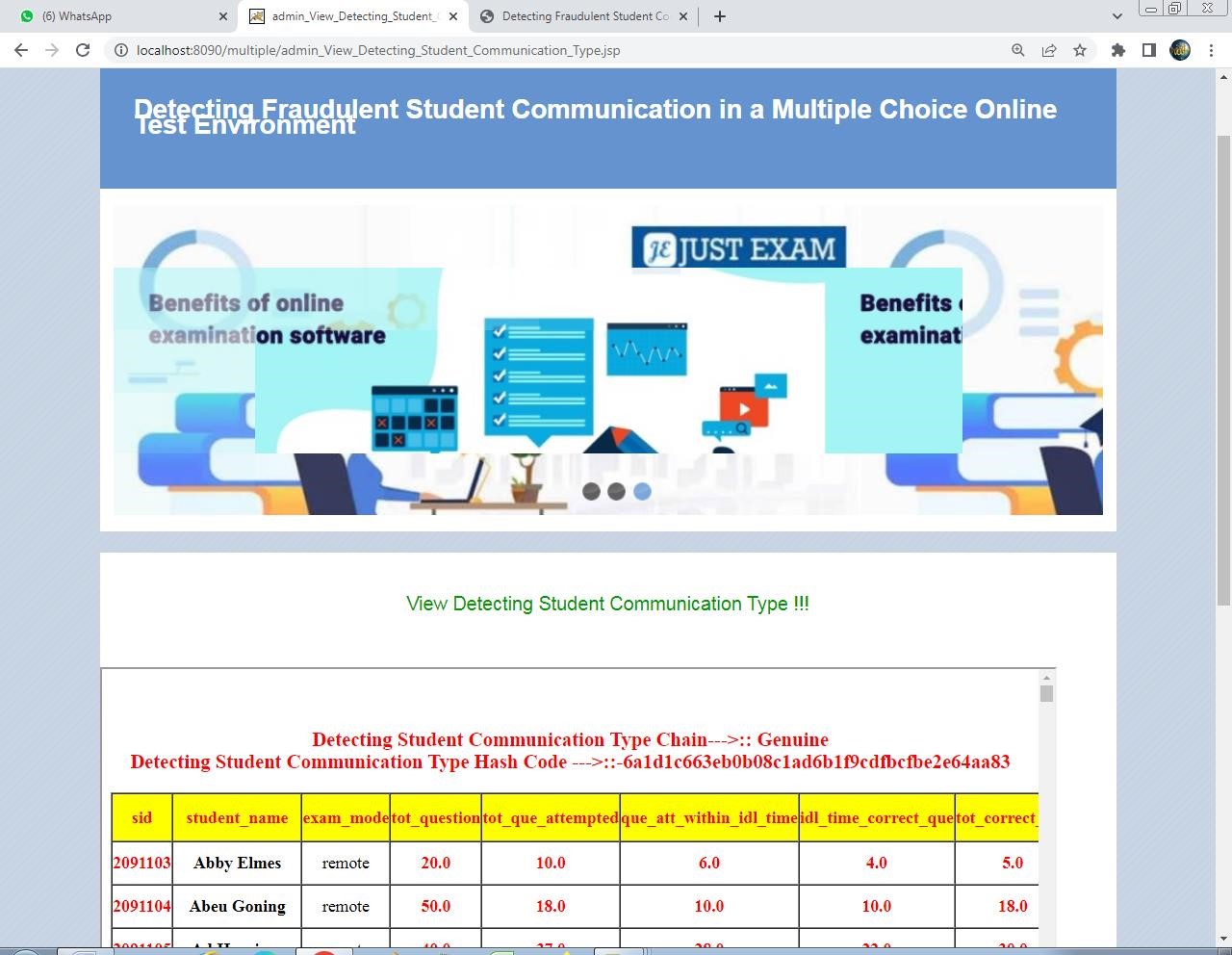
**Screen 5.1.5:** View Datasets



Screen 5.1.5: View Datasets

**Description:** In the above screen showsadmin can view the datasets.

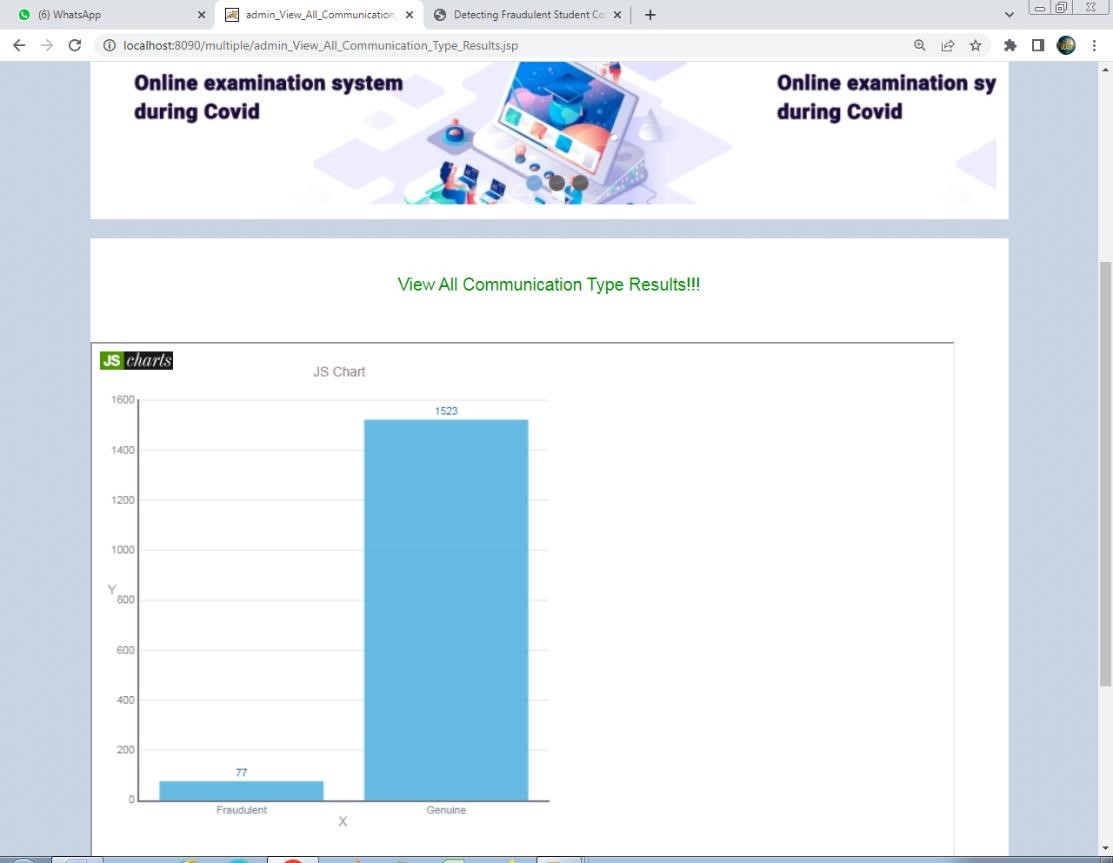
**Screen 5.1.6:** View Communication Type Chain



**Screen 5.1.6:** View Communication Type Chain

**Description:** In the above screen shows the student communication type.

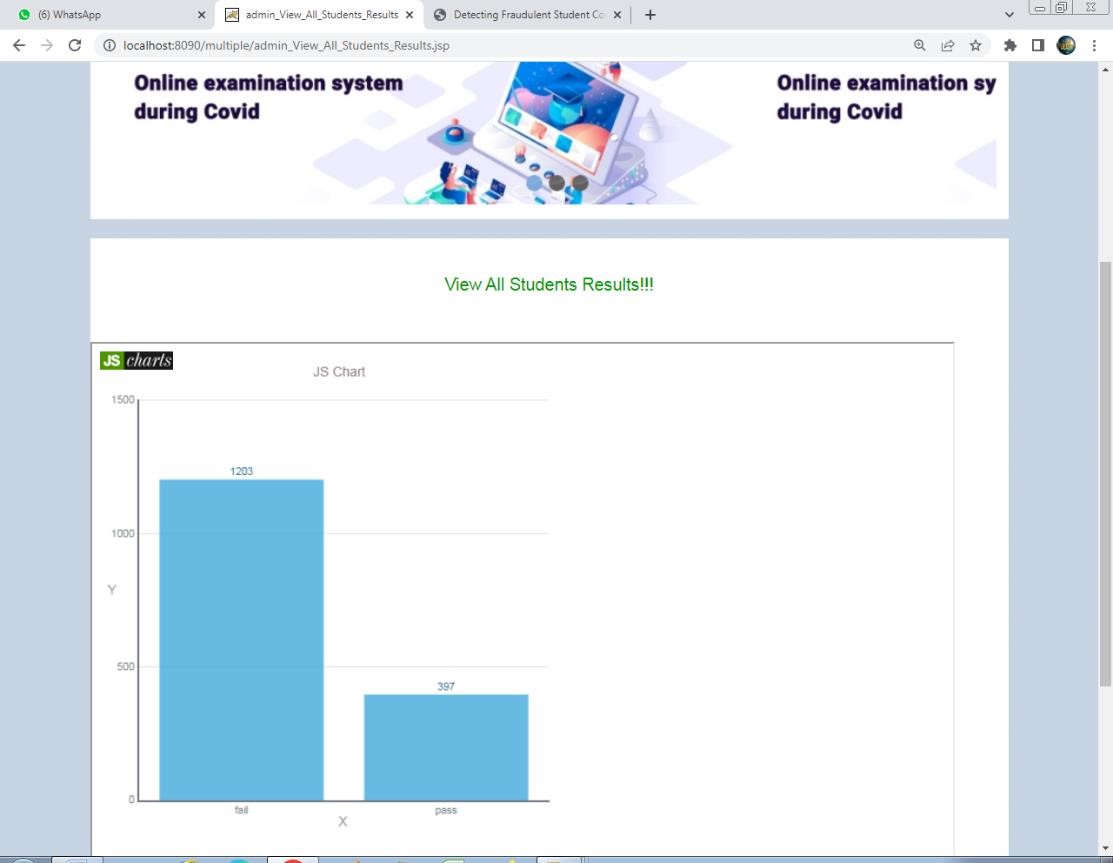
Screen 5.1.7: Graph View



Screen 5.1.7: Graph View

**Description:** In the above screen the graph shows the view of all communication type results

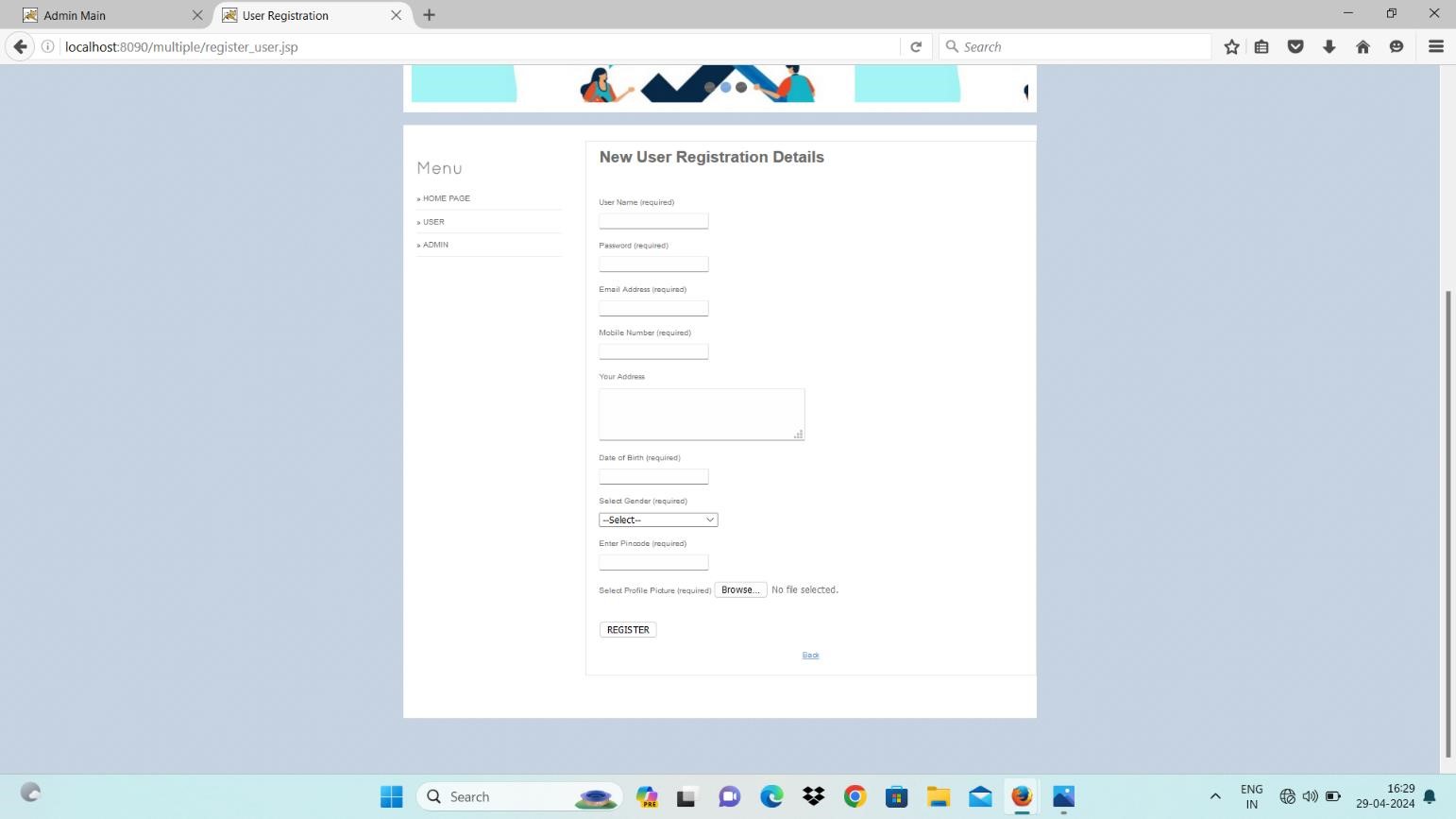
Screen 5.1.8: Graph View



Screen 5.1.8: Graph View

**Description:** In the above screen it shows pass percentage and fail percentage of students

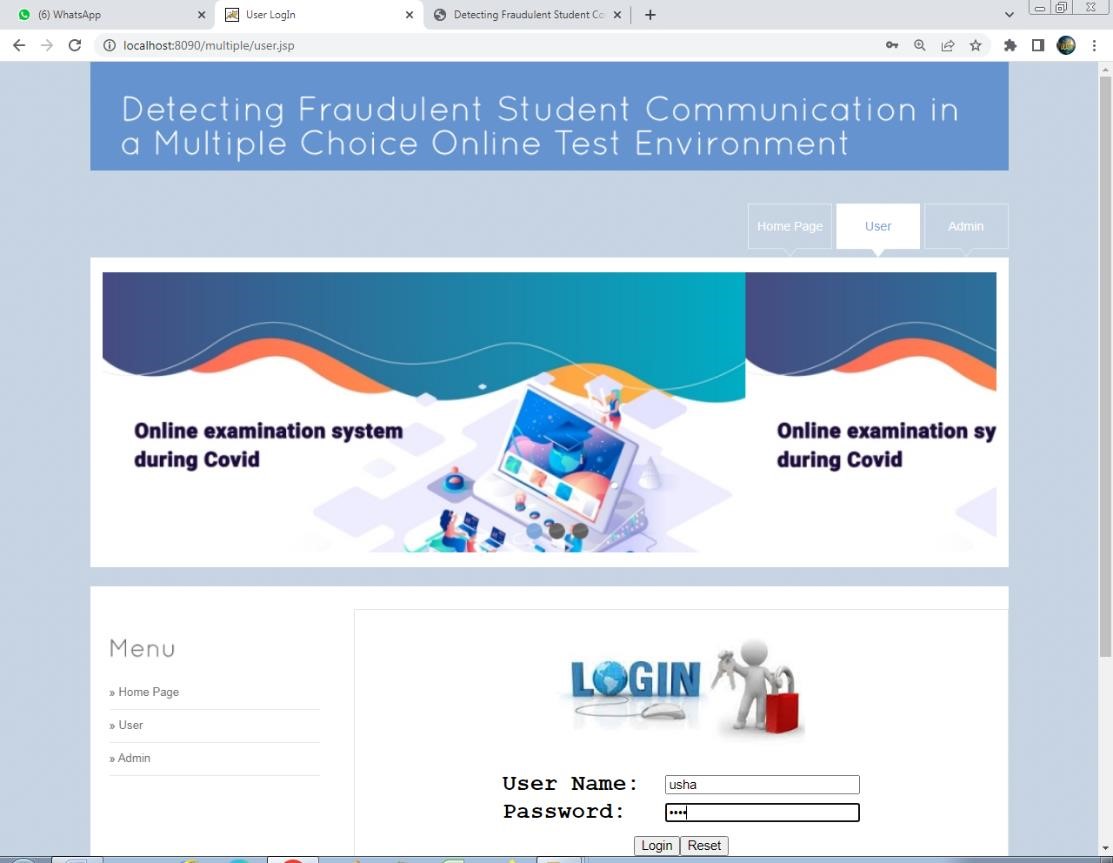
**Screen 5.1.9:** User Registration



**Screen 5.1.9:** User Registration

**Description:** In the above screen it shows registration form for user

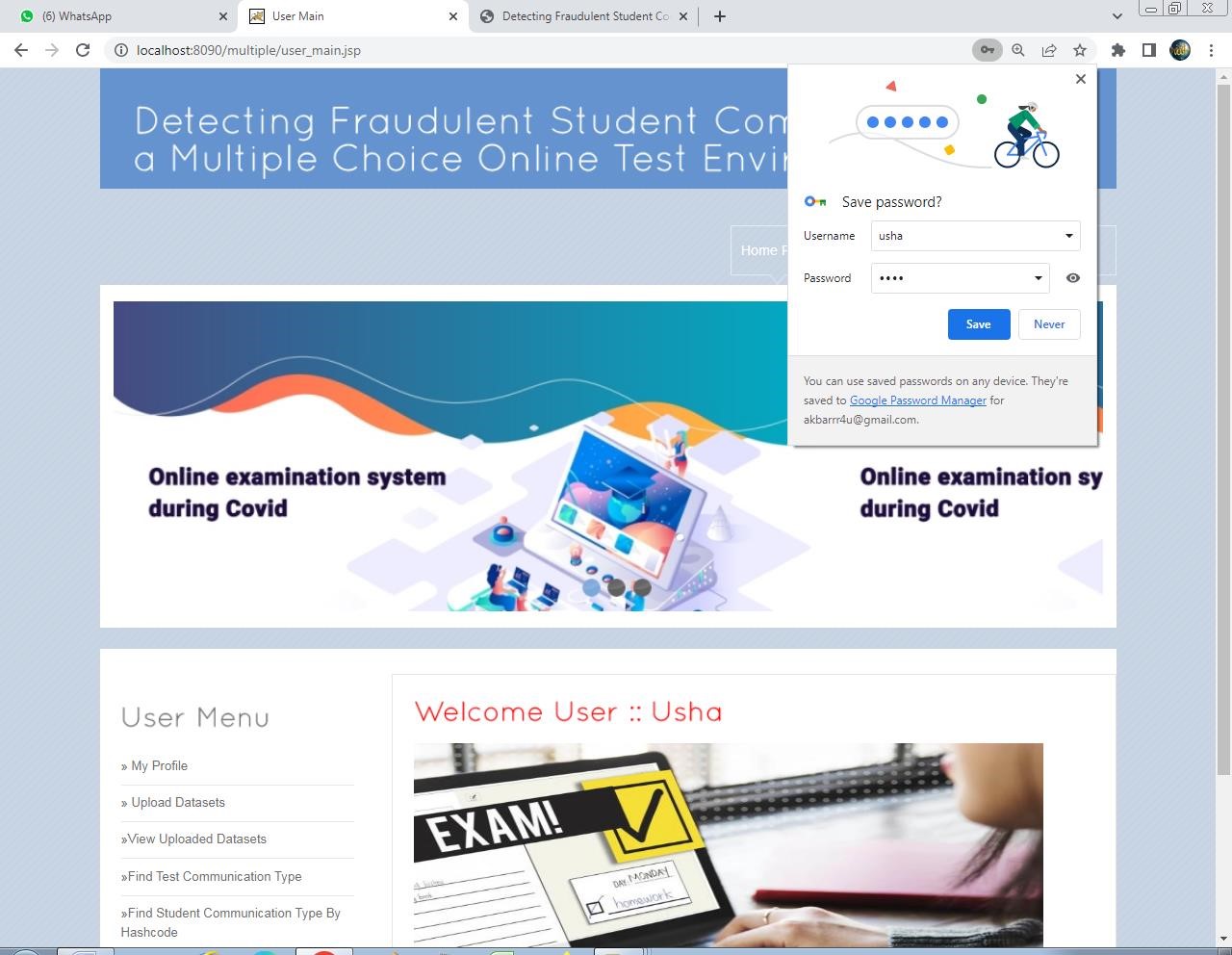
Screen 5.1.10: User Login



**Screen 5.1.10:** User Login

**Description:** In the above screen it shows the login menu

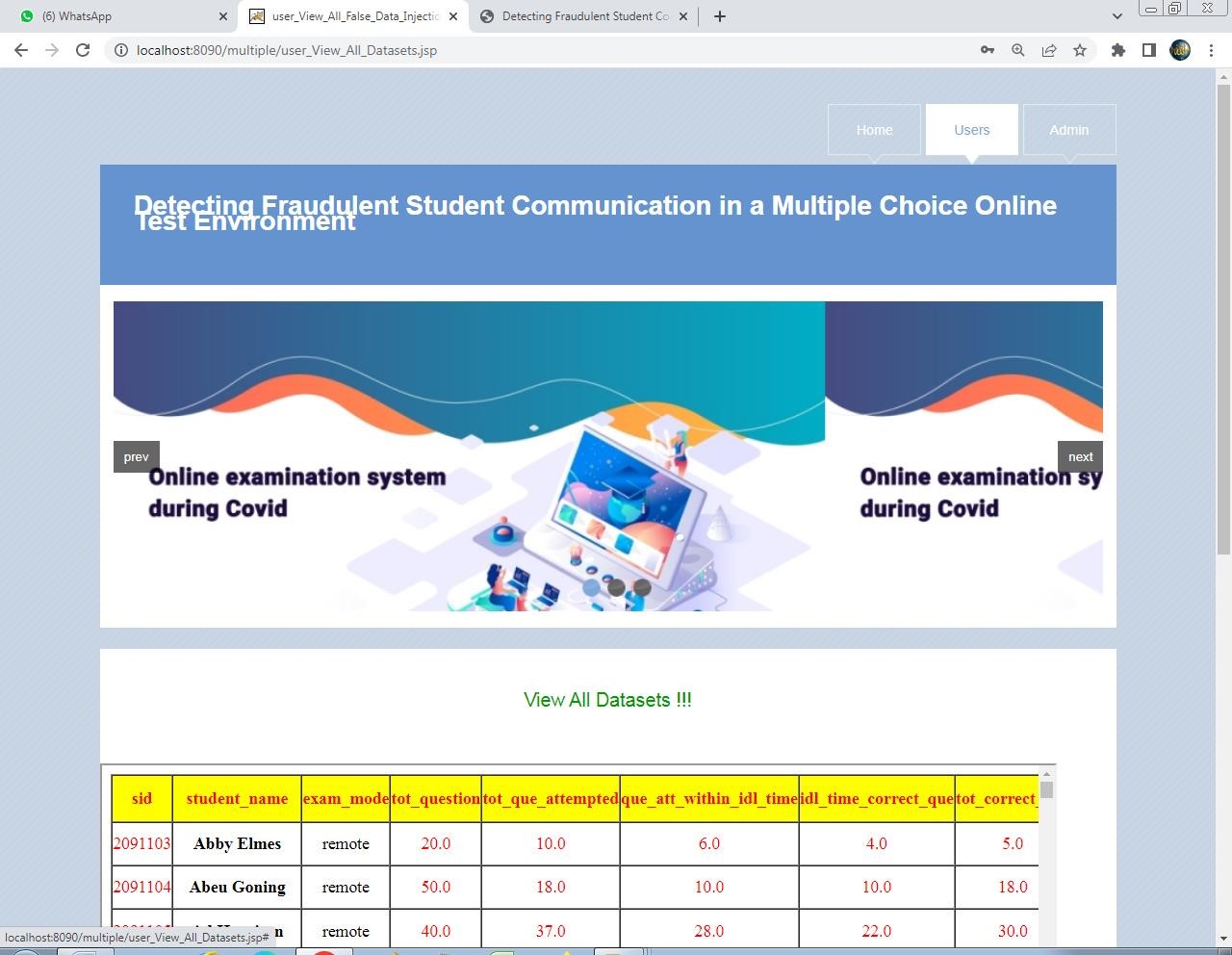
Screen 5.1.11: User Menu



Screen 5.1.11: User Menu

**Description:** In the above screen it shows login credentials for user

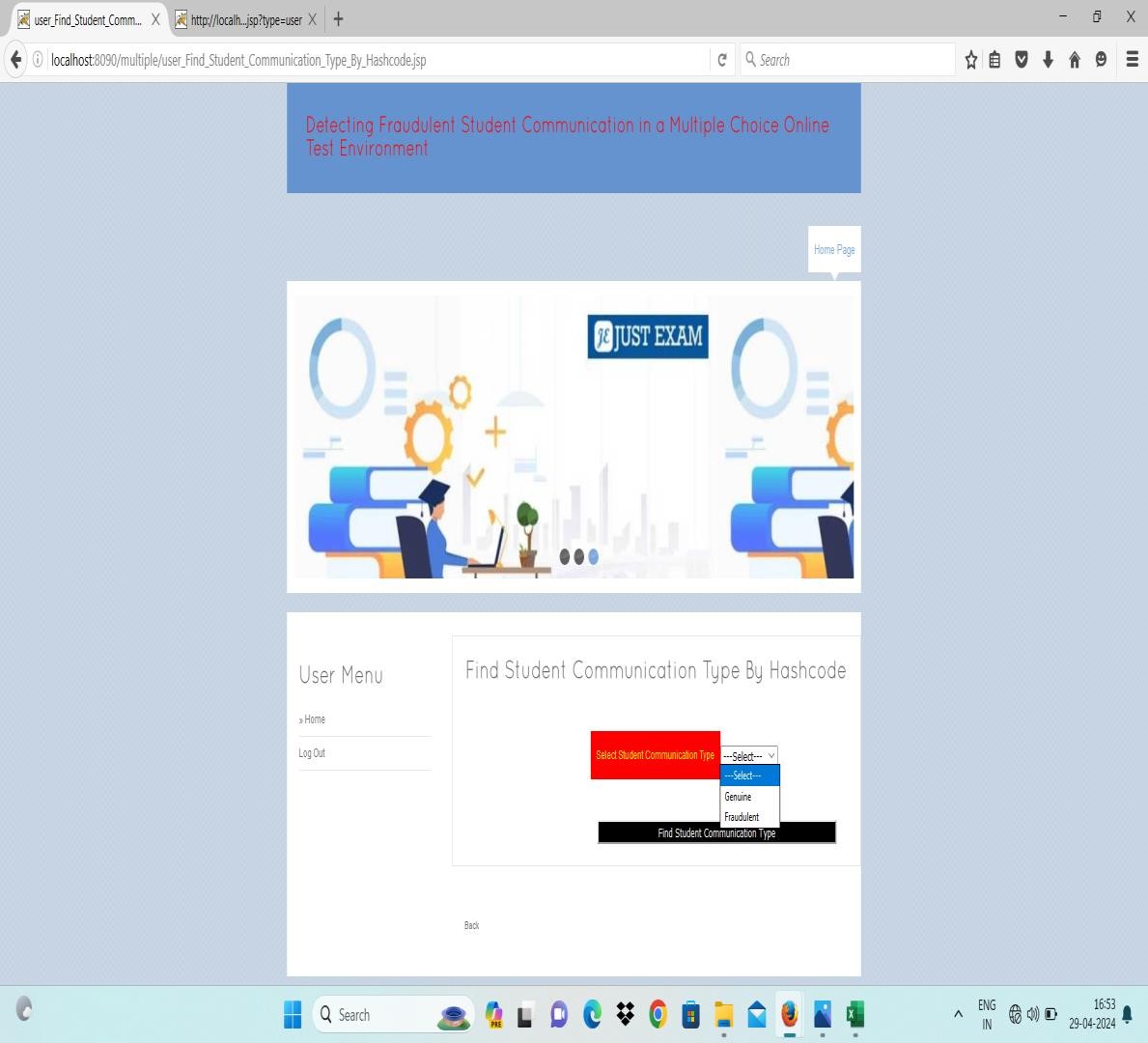
**Screen 5.1.12**: User Results



**Screen 5.1.12**: User Results

**Description:** In the above screen it shows the view of all results

**Screen 5.1.13:** Find Test Communication Type



**Screen 5.1.13:** Find Test Communication Type

**Description:** In the above screen it shows Test Communication Type

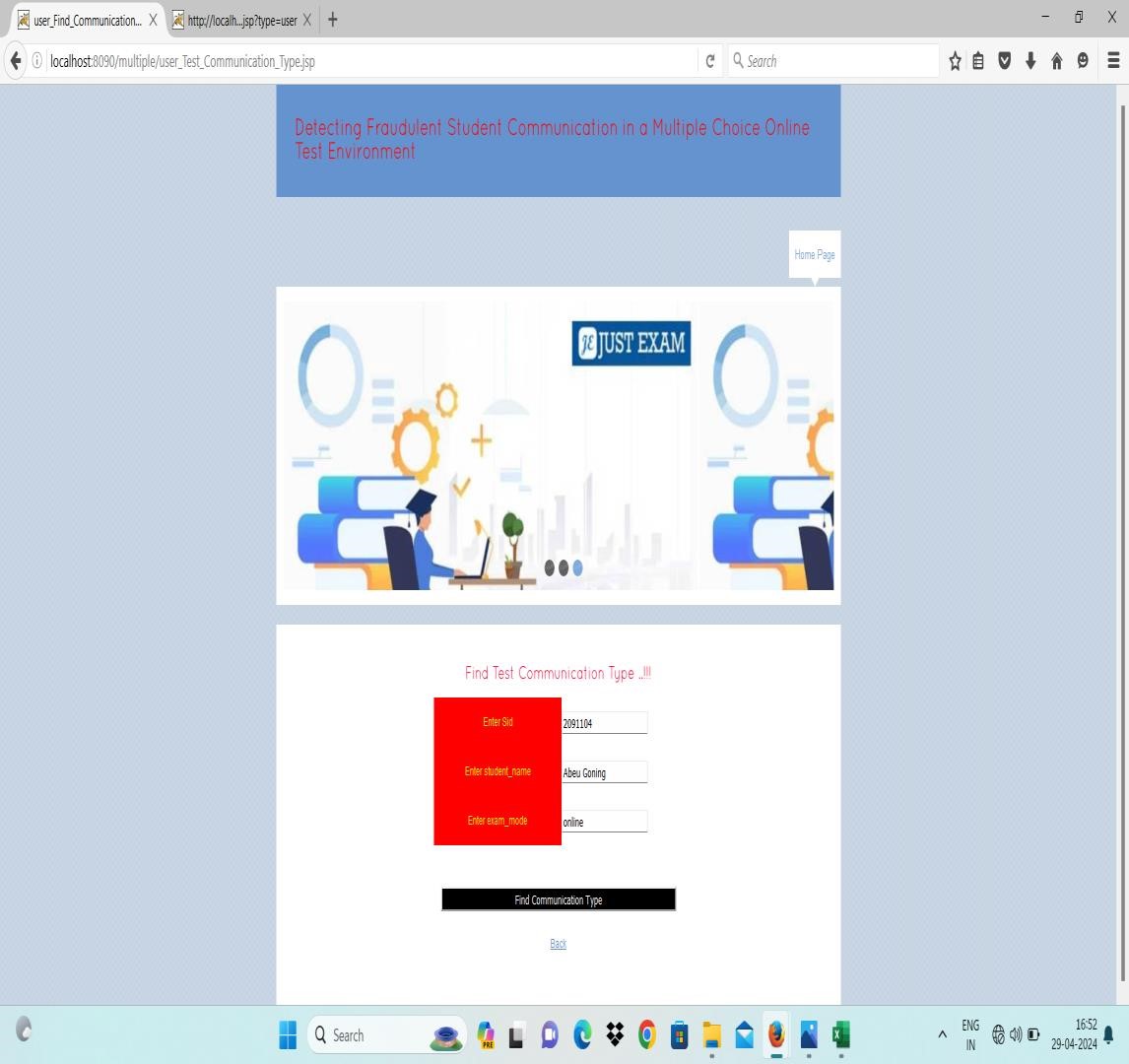
Screen 5.1.14: User Results



Screen 5.1.14: User Results

**Description:** In the above screen it shows user results

**Screen 5.1.15:** Find test communication type



**Screen 5.1.15:** Find test communication type

**Description:** In the above screen it shows whether the student is fraud or genuine

**CONCLUSION**

**CONCLUSION**

This work introduced a novel methodology to assess likely fraud communication acts in remote online MCQ exams based on the concordance of responses and answer times. Null models are produced to understand regular versus fraud dynamics and to identify collusion with strict guarantees of statistical significance. Complementarily, clustering algorithms are applied to unravel communication channels between students. Considering matched answers, choice probability, response times (directionality), and recurrence, we show that is possible to create a network of potential communication acts between students. Having constructed the network for null models representing fraudulent and honest behavior, we obtain insights into how to separate spurious communication from the actual interchange of information. Finally, employing these insights on the real data, and making use of scoring techniques, we are able to categorize each student with respect to their fraud likelihood and thus understand inadvertent communication pathways and promote the actionability of recommendations, supporting the course’s tutor with the subsequent inquiry or advertence initiatives. The application of the proposed principles in the context of the SA course reveals students with a higher fraud likelihood, already showing to be a solid criterion to guide tutors in ascertaining collusion and discouraging communication. In this work, fraudulent behavior analysis was primarily pursued in the context of a single quiz. However, if deviant behavior is detected in more than one quiz, the chances of fraudulent behavior considerably increase. In this context, binomial testing can be straightforwardly applied to identify the probability of observing a given number of potential fraud acts.

**BIBLIOGRAPHY**

**Bibliography**

**Appendix-A**

###### URL Listing

[**www.w3schols.com**](http://www.w3schols.com/)

[**www.google.co.in**](http://www.google.co.in/)

**References**

1. A. M. Duhaim, S. O. Al-Mamory, and M. S. Mahdi, “Cheating detec- tion in online exams during COVID-19 pandemic using data mining techniques,” *Webology*, vol. 19, pp. 1–26, 2021.

1. R. Bawarith, D. Abdullah, D. Anas, and S. Gamalel-Din, “E-exam cheating detection system,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 4,pp. 176–181, 2017.

1. B. A. Barr and S. F. Miller, “Higher education: The online teaching and learning experience,” Fac. School Adv. Stud., ERIC, Univ. Phoenix, Phoenix, AZ, USA, Tech. Rep. ED543912, 2013.

1. R. Matos and J. Barber, “MoodleGate: Securing computer driven exam environments,” in *Proc. INTED*, 2013, pp. 1–7.

1. V. Susithra, A. Reshma, B . Gope, and S. Sankar, “Detection of anomalous behavior in online exam towards automated proctoring,” in Proc.Int.Conf.Syst., Comput., Autom.Netw.(ICSCAN).Jul.2021, pp. 1-5.

1. G. R. Watson and J. Sottile, “Cheating in the digital age: Do students cheat more in online courses?” *Online J. Distance Learn. Admin.*, vol. 13, no. 1, pp. 1– 14, 2010.

1. C. M. Toquero, “Challenges and opportunities for higher education amid the

COVID-19 pandemic: The philippine context,” *Pedagogical Res.*, vol. 5, no. 4, Apr.

2020, Art. no. em0063.

1. F. Kamalov, H. Sulieman, and D. S. Calonge, “Machine learning based approach to exam cheating detection,” *PLoS ONE*, vol. 16, no. 8, Aug. 2021, Art. no. e0254340.

1. P. R. Morales and M. L. Verde, “Cómo asegurar evaluaciones válidas y detectar falseamiento en pruebas a distancia síncronas,” *Revista Digit. de Investigación en Docencia Universitaria*, vol. 14, no. 2, p. e1240, Nov. 2020.

1. R. K. Ladyshewsky, “Post-graduate student performance in ‘supervised inclass’ vs. ‘unsupervised online’ multiple choice tests: Implications for cheating and test security,” *Assessment Eval. Higher Educ.*, vol. 40, no. 7, pp. 883–897, Oct.

2015.

1. Z. Zhang, S. Zhu, J. Mink, A. Xiong, L. Song, and G. Wang, “Beyond bot detection: Combating fraudulent online survey takers,” in *Proc. ACM Web Conf.*, Apr. 2022, pp. 699–709.

1. H. Hu, Z. Li, and Z. Wang, “Test cheating detection method based on random forest,” in *Proc. 3rd Int. Conf. Comput. Sci. Technol. Educ. (CSTE)*, May 2021, pp.

47–52.

1. J. Ranger, N. Schmidt, and A. Wolgast, “The detection of cheating on Eexams in higher education—The performance of several old and some new indicators,” *Frontiers Psychol.*, vol. 11, Oct. 2020, Art. no. 568825.

1. M. Li et al., “Optimized collusion prevention for online exams during social distancing,” *NPJ Sci. Learn.*, vol. 6, no. 1, pp. 1–9, Mar. 2021.

1. G. J. Cizek and J. A. Wollack, *Handbook of Quantitative Meth- ods for Detecting Cheating on Tests*. Evanston, IL, USA: Routledge, 2017.

**Appendix-B**

###### Glossary

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Abbreviation** | **Description** |
| 1 | SRS | Software Requirement Specification |
| 2 | SDLC | Software Development Life Cycle |
| 3 | UML | Unified Modeling Language |
| 4 | JSP | Java Server Page |
| 5 | SQL | Structured Query Language |
| 6 | JVM | Java Virtual Machine |
| 7 | HTML | Hyper Text Markup Language |
| 8 | ODBC | Open Database Connectivity |
| 9 | JDBC | Java Database Connectivity |

**Appendix – C**

**List of Figures**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Fig No.** | **Description** | **Page No.** | **Chapter** |
| 1 | 1.4 | System Architecture | 5 | Introduction |
| 2 | 2.1 | Product Perspective | 6 | SRS |
| 3 | 2.2 | Characteristics of Product Perspective | 7 | SRS |
| 4 | 2.8.1 | SDLC | 12 | SRS |
| 5 | 2.8.2 | Waterfall Model | 14 | SRS |
| 6 | 2.10.1 | Working of Java Program | 17 | SRS |
| 7 | 2.10.2 | Implementation of Java Virtual Machine | 18 | SRS |
| 8 | 2.10.3 | Program Running on the  Java Platform | 19 | SRS |
| 9 | 2.10.4 | Properties wizard | 21 | SRS |
| 10 | 2.10.5 | Path setting for java | 21 | SRS |
| 11 | 2.10.6 | Welcome page of tomcat | 22 | SRS |
| 12 | 2.10.7 | Tomcat configuration option page | 22 | SRS |
| 13 | 2.10.8 | Tomcat license agreement | 23 | SRS |
| 14 | 2.10.9 | Welcome wizard of MySQL | 24 | SRS |
| 15 | 2.10.10 | SQL setup wizard | 24 | SRS |
| 16 | 2.10.11 | Database configuration engine | 25 | SRS |
| 17 | 2.10.12 | Tomcat Page | 25 | SRS |
| 18 | 3.1.1 | ER-Model | 27 | System Design |
| 19 | 3.1.2 | Components of ER Diagram | 27 | System Design |
| 20 | 3.1.3 | E-R Diagram | 32 | System Design |
| 21 | 3.4.1 | UML Diagram | 49 | System Design |
| 22 | 3.4.2 | Use case Diagram | 50 | System Design |
| 23 | 3.4.3 | Class Diagram | 51 | System Design |
| 24 | 3.4.4 | Sequence Diagram | 52 | System Design |
| 25 | 3.4.5 | Activity Diagram | 53 | System Design |
| 26 | 3.4.6 | Component Diagram | 54 | System Design |
| 27 | 3.4.7 | Deployment Diagram | 55 | System Design |
| 28 | 3.4.8 | Collaboration Diagram | 56 | System Design |
| 29 | 4.1 | Levels of Testing | 58 | Testing |

**LIST OF TABLES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Table No.** | **Table Name** | **Page** **No.** | **Chapter** |
| 1 | 3.2.1 | Data Dictionary | 33 | System Design |
| 2 | 3.2.2 | Data Table | 34 | System Design |
| 3 | 3.2.3 | Received User Data Analysis | 34 | System Design |
| 4 | 3.3.1 | 1NF | 35 | System Design |
| 5 | 3.3.2 | 2NF | 36 | System design |
| 6 | 3.3.3 | User Details | 36 | System Design |
| 7 | 3.3.4 | User Details | 36 | System Design |
| 8 | 3.3.5 | Normal Forms | 38 | System Design |
| 9 | 4.4 | Test Cases | 63 | Testing |

**LIST OF SCREENS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Screen.No** | **Screen Name** | **Page No.** | **Chapter** |
| 1 | 5.1.1 | Home Page | 64 | Implementation |
| 2 | 5.1.2 | Admin Login Page | 65 | Implementation |
| 3 | 5.1.3 | Admin Menu Page | 66 | Implementation |
| 4 | 5.1.4 | View and Authorize | 67 | Implementation |
| 5 | 5.1.5 | View Datasets | 68 | Implementation |
| 6 | 5.1.6 | View Communication type Chain | 69 | Implementation |
| 7 | 5.1.7 | Graph View | 70 | Implementation |
| 8 | 5.1.8 | Graph View | 71 | Implementation |
| 9 | 5.1.9 | User Registration | 72 | Implementation |
| 10 | 5.1.10 | User Login | 73 | Implementation |
| 11 | 5.1.11 | User Menu | 74 | Implementation |
| 12 | 5.1.12 | User Result | 75 | Implementation |
| 13 | 5.1.13 | Find Test Communication Type | 76 | Implementation |
| 14 | 5.1.14 | User Results | 77 | Implementation |
| 15 | 5.1.15 | Find Test Communication Type | 78 | Implementation |

**APPENDIX - D**

**BASE PAPER**