

STUDENT DATABASE MANAGEMENT SYSTEM

INTERDISCIPLINARY PROJECT

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in Information Technology

by

Avinash Kumar Gupta (41120034)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A++” by NAAC

12B Status by UGC | Approved by AICTE

JEPPIAR NAGAR, RAJIV GANDHISALAI, CHENNAI - 600119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Professional Training is the bonafide work of **MR.AVINASH KUMAR GUPTA (41120034)** who carried out the project entitled **"STUDENT DATABASE MANAGEMENT SYSTEM"** under my supervision from **January 2024 to May 2024.**

Internal Guide
Dr. S. URMELA, M.Tech., Ph.D.,

Head of the Department
Dr. R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voice Examination held on 08/05/2024

Internal Examiner

05.

External Examiner

DECLARATION

I, **AVINASH KUMAR GUPTA (41120034)**, hereby declare that the Professional Training Report-II entitled "**STUDENT DATABASE MANAGEMENT SYSTEM**" done by me under the guidance of **DR. S. URMELA, M.Tech., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 08/05/2024

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.SASIKALA M.E., Ph.D., Dean**, School of Computing, **Dr. R. SUBHASHINI, M.E., Ph.D., Head Of The Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. S. URMELA, M.Tech., Ph.D.**, for his/her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-1 professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department Of Information Technology** who were helpful in many ways for the completion of the project.

CERTIFICATE

	Elite NPTEL Online Certification (Funded by the MoE, Govt. of India) This certificate is awarded to AVINASH KUMAR GUPTA for successfully completing the course Data Base Management System with a consolidated score of 67 % <table><tr><td>Online Assignments</td><td>22.5/25</td><td>Proctored Exam</td><td>44.25/75</td></tr></table> Total number of candidates certified in this course: 6225 Jan-Mar 2024 (8 week course)	Online Assignments	22.5/25	Proctored Exam	44.25/75	  Prof. Haimanti Banerji Coordinator, NPTEL IIT Kharagpur
Online Assignments	22.5/25	Proctored Exam	44.25/75			
 Indian Institute of Technology Kharagpur						
Roll No: NPTEL24CS21S544105750	To verify the certificate 	No. of credits recommended: 2 or 3				

ABSTRACT

The Student Database Management System (SDBMS) is a Java-based application designed to facilitate the registration and management of student data in educational institutions. This project integrates a graphical user interface (GUI) for seamless student registration and data viewing, ensuring a user-friendly experience.

One of the key features of SDBMS is its client-server architecture, enhancing security and scalability. The client component allows users to input student information, which is then encrypted before being transmitted to the server. This encryption ensures the confidentiality and integrity of the data during transmission. On the server side, the application utilizes MySQL database to securely store the student information. The server decrypts the received data, validates it, and then stores it in the database. Additionally, when requested by the client, the server retrieves the data from the database, encrypts it, and sends it back to the client. This encryption ensures that sensitive student information remains protected both in transit and at rest.

The SDBMS not only provides a robust solution for managing student data but also prioritizes security by implementing encryption mechanisms throughout the data transmission and storage processes. With its intuitive GUI, client-server architecture, and emphasis on data security, SDBMS offers an efficient and secure solution for educational institutions to manage student information effectively.

CONTENTS

Ch No.	TITLE	Page No.
	ABSTRACT	vi
	LIST OF ABBREVIATIONS	ix
	LIST OF FIGURES	x
1	INTRODUCTION	1
1.1	Overview	1
1.1.1	Student Data	1
1.1.2	Ngrok	2
1.1.3	Java Networking	3
1.1.4	Server And Client Communication	4
1.1.5	Database	6
1.2	Literature Survey	8
1.2.1	Evolution Of Student Database Management Systems	8
1.2.2	Challenges In Student Database Management	8
1.2.3	Innovation And Best Practice In SDMS	8
1.2.4	Trends In SDMS Research	9
1.2.5	Future Directions And Recommendations	9
2	METHODOLOGY	10
2.1	Methodology	10
2.1.1	Introduction	10
2.1.2	Requirement Gathering	10
2.1.3	System Design	10
2.1.4	Database Development	11

2.1.5	Implementation	11
2.1.6	Testing	11
2.1.7	Deployment	11
2.1.8	Maintainence	11
2.2	Architecture	12
2.2.1	User Interface	13
2.2.2	Business Logic	13
2.2.3	Data Management	13
2.2.4	Security	14
2.2.5	Architecture Of NGROK TCP Tunelling	15
2.3	Modules	16
2.3.1	Ngrok	16
2.3.2	Java.NET Package	18
2.3.3	Java.SWING Package	19
2.4	Algorithm	21
3	RESULTS AND DISCUSSION	23
4	CONCLUSION AND FUTURE WORK	26
4.1	Conclusion	26
4.2	Future Work	26
5	REFERENCES	27
6	APPENDICES	29
A.	Sample Code	29
B.	Screenshots	33

LIST OF ABBREVIATIONS

ABBREVIATION		EXPANSION
URL	-	Uniform Resource Locator
TSL	-	Transport Layer Security
SSL	-	Secure Sockets Layer
NAT	-	Network Address Translation
TCP	-	Transmission Control Protocol
HTTP	-	Hypertext Transfer Protocol
FTP	-	File Transfer Protocol
UDP	-	User Datagram Protocol
SQL	-	Structured Query Language
API	-	Application Programming Interface
RDS	-	Relational Database Service
SDMS	-	Student Database Management System
FERPA	-	Family Education Rights & Privacy Act
GDPR	-	General Data Protection Regulation
SIS	-	Student Information System
LMS	-	Learning Management System
IOT	-	Internet Of Things
RDBMS	-	Relational Database Management System
AI	-	Artificial Intelligence
NLP	-	Natural Language Processing

LIST OF FIGURES

FIGURE NO.	NAME OF FIGURE	PAGE NO.
1.1	Client Server Communication	6
2.1	SDMS Architecture	12
2.2	NGROK Tunnel	14
3.1	Data Entry Panel	23
3.2	Data View Panel	24

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

A student database management system (SDMS) is a computerized system that helps educational institutions store, maintain, and process student data. It's also known as a student information system or student management system .

1.1.1 STUDENT DATA

Student data encompasses a wide array of information crucial for facilitating educational processes and enhancing student outcomes. This includes personal details such as names, addresses, and contact information, as well as academic records, attendance records, disciplinary history, and special needs accommodations. Analyzing this data provides educators and administrators with insights into individual student performance, learning patterns, and overall trends within the student body. By leveraging student data effectively, institutions can tailor instruction, identify at-risk students, allocate resources efficiently, and foster a supportive learning environment.

While student data offers immense potential, managing it presents a host of challenges. Data security and privacy are paramount concerns, with institutions obligated to safeguard sensitive information against unauthorized access, breaches, and misuse. Compliance with data protection regulations such as the Family Educational Rights and Privacy Act (FERPA) in the United States and the General Data Protection Regulation (GDPR) in Europe adds complexity to data management practices. Additionally, ensuring data accuracy, consistency, and accessibility across various systems and platforms poses logistical hurdles for educational institutions.

Advancements in technology have transformed student data management, offering innovative solutions to streamline administrative processes and enhance data-driven decision-making. Modern Student Information Systems (SIS) and Learning Management Systems (LMS) integrate robust features for student data collection, analysis, and reporting. Cloud-based solutions provide scalability, flexibility, and centralized data storage, enabling real-time access to information

from anywhere, at any time. Furthermore, the emergence of data analytics tools empowers educators to extract actionable insights from vast datasets, facilitating personalized learning experiences and targeted interventions.

In an era defined by digital innovation and data-driven insights, student data management emerges as a cornerstone of educational excellence. By harnessing the power of data responsibly, educational institutions can nurture student success, foster inclusive learning environments, and adapt to the evolving needs of 21st-century learners. As we navigate the complexities of data management, a commitment to data integrity, privacy, and ethical use remains paramount, ensuring that student data continues to serve as a catalyst for positive educational outcomes.

1.1.2 NGROK

Ngrok is a powerful tool used for creating secure tunnels to localhost. It enables developers to expose local servers behind NATs and firewalls to the public internet over secure tunnels. With Ngrok, developers can easily test webhooks, APIs, and web applications locally without deploying them to a public server.

One of the key features of Ngrok is its simplicity. Developers can start using Ngrok with just a single command, making it ideal for quick prototyping and development. By running `ngrok http`, developers can instantly create a tunnel to their local server and receive a unique URL accessible from anywhere.

Ngrok provides secure connections by default, encrypting all traffic between the local server and the public internet using TLS/SSL. This ensures that sensitive data transmitted through the tunnel remains protected from eavesdropping and tampering.

In addition to basic tunneling, Ngrok offers advanced features such as custom subdomains, TCP tunneling, and HTTP request inspection. Developers can configure custom subdomains to create more memorable URLs for their tunnels, while TCP tunneling allows for tunneling of non-HTTP protocols such as SSH and FTP.

Furthermore, Ngrok provides detailed logging and inspection capabilities, allowing developers to monitor traffic passing through their tunnels in real-time. The web-based dashboard provides insights into request and response headers, status codes, and latency, enabling developers to diagnose issues and optimize performance.

Ngrok offers both free and paid plans, with the free plan providing basic tunneling functionality and the paid plans offering additional features such as custom domains, reserved TCP addresses, and increased tunneling capacity.

Overall, Ngrok is a versatile tool that simplifies the process of exposing local servers to the internet securely. Its ease of use, robust security features, and advanced capabilities make it an essential tool for developers working on web development, API testing, and debugging tasks.

1.1.3 JAVA NETWORKING

Java networking offers a vast array of tools and functionalities for building connected applications that communicate over networks. At its core, Java's networking capabilities are rooted in the `java.net` package, which provides classes and interfaces for working with network connections, sockets, URLs, and more. This package serves as the foundation for creating robust and scalable networking solutions in Java.

One of the fundamental concepts in Java networking is socket programming. Sockets facilitate communication between two endpoints over a network. The `Socket` class represents an endpoint for communication, providing input and output streams for sending and receiving data. Conversely, the `ServerSocket` class listens for incoming connections from clients and creates sockets for communication upon acceptance. This client-server model forms the basis for many networked applications, allowing clients to send requests to servers and receive responses in return.

In addition to socket programming, Java networking also encompasses URL handling. The `URL` class in Java provides a convenient way to work with URLs, allowing developers to retrieve data from web resources. With URLs, applications can open connections to web servers, download files, and interact with web APIs

seamlessly. `URLConnection`, another class in Java's networking arsenal, enables developers to establish connections to URLs and perform various operations such as reading from and writing to the resource represented by the URL.

Networking in Java extends beyond basic communication to support various networking protocols such as TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP offers reliable, connection-oriented communication, making it suitable for applications where data integrity is crucial. On the other hand, UDP provides lightweight, connectionless communication, ideal for applications that prioritize speed and efficiency over reliability.

Asynchronous networking is another area where Java shines. The `java.nio` package provides support for non-blocking I/O operations, allowing applications to handle multiple connections concurrently without blocking threads. This asynchronous networking capability improves scalability and performance, making it well-suited for high-throughput applications.

Security considerations are paramount in networking applications, and Java provides robust mechanisms to ensure secure communication. SSL/TLS protocols can be used for encrypted communication and authentication, safeguarding sensitive data from unauthorized access and eavesdropping. Developers must also be vigilant against potential security vulnerabilities such as injection attacks and implement appropriate safeguards to mitigate risks.

In conclusion, Java networking empowers developers to build powerful and scalable applications that communicate seamlessly over networks. By leveraging Java's rich set of networking tools and protocols, developers can create a wide range of networked applications, from client-server systems to web APIs, while ensuring security and reliability.

1.1.4 SERVER & CLIENT COMMUNICATION

Server and client communication is a fundamental aspect of networked applications, enabling the exchange of data and services between multiple devices over a network. This communication paradigm follows a client-server model, where one device (the server) provides resources or services, while other devices (clients) request and utilize these resources.

In server-client communication, the server typically listens for incoming connections on a designated network port, while clients establish connections to the server to request or send data. The server responds to client requests, processes them, and returns the requested data or performs the requested actions.

The choice of programming language for implementing server and client applications depends on various factors, including performance requirements, scalability, and developer preferences. Languages such as C++, Java, Python, and JavaScript are commonly used for building server and client applications, each offering unique features and capabilities.

Server applications handle tasks such as data processing, resource management, and authentication, while client applications interact with users and facilitate communication with the server. Communication between servers and clients typically occurs over network protocols such as TCP/IP, UDP, HTTP, or WebSocket, depending on the specific requirements of the application.

Several frameworks and libraries simplify the development of server and client applications in different programming languages. For example, in Java, developers can use frameworks like Spring Boot or Netty for building robust server applications, while libraries like OkHttp and Retrofit facilitate client-side networking tasks. Similarly, in Python, frameworks like Flask and Django provide tools for building server applications, while libraries like requests and aiohttp offer client-side networking capabilities.

Server and client communication is essential for a wide range of applications, including web services, mobile apps, multiplayer games, and IoT devices. By implementing efficient and reliable communication protocols and leveraging appropriate programming languages and frameworks, developers can build scalable, secure, and high-performance server-client applications to meet diverse user needs and requirements.

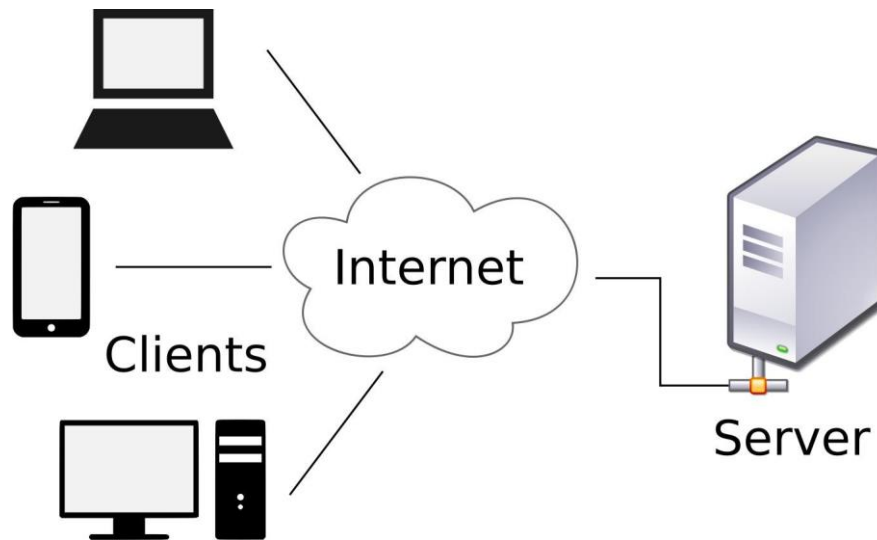


Fig 1.1 Client Server Communication

1.1.5 DATABASE

Database management is a critical aspect of modern information systems, enabling organizations to efficiently store, retrieve, and manage vast amounts of data. Databases serve as centralized repositories for organizing structured and unstructured data, providing mechanisms for data storage, retrieval, manipulation, and analysis.

In database management, structured query languages (SQL) such as SQL, MySQL, PostgreSQL, and Oracle are commonly used to interact with databases. These languages enable developers to perform various operations, including creating databases and tables, inserting, updating, and deleting data, and querying data using advanced filtering and aggregation techniques.

Database management systems (DBMS) play a central role in database management, providing software tools and services for creating, managing, and accessing databases. DBMS solutions come in various forms, including relational database management systems (RDBMS), NoSQL databases, and cloud-based

database services, each offering different features and capabilities to meet specific application requirements.

Relational database management systems (RDBMS) such as MySQL, PostgreSQL, and Microsoft SQL Server are widely used for managing structured data in relational tables. These systems enforce data integrity through primary key and foreign key constraints, support transactions for ensuring data consistency, and offer powerful querying capabilities through SQL.

NoSQL databases such as MongoDB, Cassandra, and Redis are designed for managing unstructured or semi-structured data, providing flexible data models and horizontal scalability to handle large volumes of data and high-throughput workloads. These databases are suitable for use cases such as real-time analytics, content management, and caching.

Cloud-based database services such as Amazon RDS, Google Cloud SQL, and Microsoft Azure Database offer scalable and managed database solutions in the cloud. These services provide features such as automated backups, high availability, and multi-region replication, allowing organizations to deploy and manage databases with minimal administrative overhead.

Database management encompasses various tasks and responsibilities, including database design, schema modeling, data migration, performance tuning, and security management. Effective database management requires careful planning, monitoring, and optimization to ensure data integrity, availability, and security while meeting performance and scalability requirements.

Overall, database management is essential for organizations to effectively manage and leverage data assets for decision-making, business intelligence, and competitive advantage. By adopting appropriate database management tools and best practices, organizations can optimize data management processes, enhance operational efficiency, and drive innovation and growth.

1.2 LITERATURE SURVEY

Student Database Management Systems (SDMS) play a pivotal role in educational institutions, facilitating the efficient storage, retrieval, and management of student-related information. This literature survey explores the evolution, challenges, and innovations in SDMS, highlighting key research findings and trends in the field.

1.2.1 EVOLUTION OF STUDENT DATABASE MANAGEMENT SYSTEMS

Early efforts in SDMS focused on manual record-keeping systems, transitioning to rudimentary digital databases in the late 20th century. With the advent of relational database management systems (RDBMS), such as MySQL and Oracle, SDMS underwent a paradigm shift towards scalable, efficient, and integrated solutions. The integration of web-based technologies further expanded the capabilities of SDMS, enabling real-time access to student data from anywhere, at any time.

1.2.2 CHALLENGES IN STUDENT DATABASE MANAGEMENT

Despite technological advancements, SDMS still faces several challenges, including data security and privacy concerns, data integration from disparate sources, and ensuring data accuracy and consistency. Research has identified the need for robust security measures to protect sensitive student information from unauthorized access and data breaches. Additionally, the interoperability of SDMS with other institutional systems remains a challenge, necessitating standardized data formats and protocols.

1.2.3 INNOVATIONS AND BEST PRACTICES IN SDMS

Recent research in SDMS has focused on innovative solutions to address these challenges and enhance the efficiency and effectiveness of student data management. Cloud-based SDMS solutions offer scalability, flexibility, and cost-effectiveness, enabling educational institutions to leverage the benefits of cloud computing while ensuring data security and compliance. Furthermore, advancements in data analytics and machine learning have enabled predictive analytics capabilities in SDMS, facilitating data-driven decision-making and personalized learning experiences for students.

1.2.4 TRENDS IN SDMS RESEARCH

Current trends in SDMS research include the integration of artificial intelligence (AI) and natural language processing (NLP) technologies to automate administrative tasks, improve data accuracy, and enhance user experiences. Additionally, the emergence of blockchain technology holds promise for enhancing the security and transparency of student data transactions, mitigating the risk of data tampering and fraud.

1.2.5 FUTURE DIRECTIONS AND RECOMMENDATIONS

Looking ahead, the future of SDMS lies in embracing emerging technologies and standards to address evolving challenges and meet the growing demands of educational institutions. Research efforts should focus on developing interoperable, scalable, and secure SDMS solutions that leverage AI, blockchain, and other cutting-edge technologies to empower educators, administrators, and students alike.

Student Database Management Systems play a critical role in modern educational institutions, facilitating efficient student data management and informed decision-making. This literature survey highlights the evolution, challenges, innovations, and future directions of SDMS research, providing valuable insights for researchers, practitioners, and policymakers in the field of education technology.

CHAPTER 2

METHODOLOGY

2.1 METHODOLOGY

The following are the important steps for the creation of a working SDBMS application:

2.1.1 INTRODUCTION

The methodology for developing a Student Database Management System (SDMS) involves a systematic approach to designing, implementing, and maintaining a robust system that efficiently manages student-related information. This comprehensive methodology encompasses various stages, including requirements gathering, system design, database development, implementation, testing, and maintenance. Below is a detailed methodology outlining each step in the development process:

2.1.2 REQUIREMENTS GATHERING

The first step in developing an SDMS is to gather requirements from stakeholders, including educators, administrators, and IT personnel. This involves conducting interviews, surveys, and workshops to identify the specific needs, functionalities, and constraints of the system. Key requirements to consider include student registration, academic records management, attendance tracking, reporting, and security.

2.1.3 SYSTEM DESIGN

Based on the gathered requirements, the next step is to design the system architecture and user interface. This involves creating high-level architectural diagrams, data flow diagrams, and user interface mockups to visualize the system's structure and functionality. The system design should incorporate principles of modularity, scalability, and usability to ensure flexibility and ease of maintenance.

2.1.4 DATABASE DEVELOPMENT

With the system architecture in place, the focus shifts to developing the database schema and data models. This involves identifying the entities, attributes, and relationships relevant to student data management and designing the database tables accordingly. Considerations should be made for data normalization, indexing, and integrity constraints to optimize performance and ensure data integrity.

2.1.5 IMPLEMENTATION

Once the database schema is finalized, the next step is to implement the SDMS according to the design specifications. This involves writing code to implement the business logic, user interface, and database interactions. Java is commonly used for developing SDMS applications, leveraging frameworks such as JavaFX for the user interface and JDBC for database connectivity.

2.1.6 TESTING

Testing is a crucial phase in the SDMS development process to ensure that the system meets the specified requirements and functions as intended. Various types of testing should be conducted, including unit testing, integration testing, system testing, and user acceptance testing. Test cases should be developed to validate each component of the system, including user interface elements, database operations, and system functionalities.

2.1.7 DEPLOYMENT

Once testing is complete and the system is deemed stable, it can be deployed to production environments for use by stakeholders. Deployment involves installing the SDMS on servers, configuring network settings, and ensuring compatibility with existing IT infrastructure. Proper documentation should accompany the deployment process to guide administrators and users in system setup and configuration.

2.1.8 MAINTENANCE

The final phase of the SDMS development methodology involves ongoing maintenance and support to ensure the system remains functional, secure, and

up-to-date. This includes monitoring system performance, addressing user feedback and bug reports, applying software updates and patches, and implementing security measures to protect against threats and vulnerabilities.

The methodology outlined above provides a structured approach to developing a Student Database Management System that meets the needs of educational institutions. By following this methodology, developers can create a robust and efficient SDMS that streamlines student data management processes, enhances data integrity and security, and supports informed decision-making by educators and administrators.

2.2 ARCHITECTURE

The functional architecture of the Student Database Management System (SDMS) project outlines the core functionalities and interactions of the system components to effectively manage student-related information.

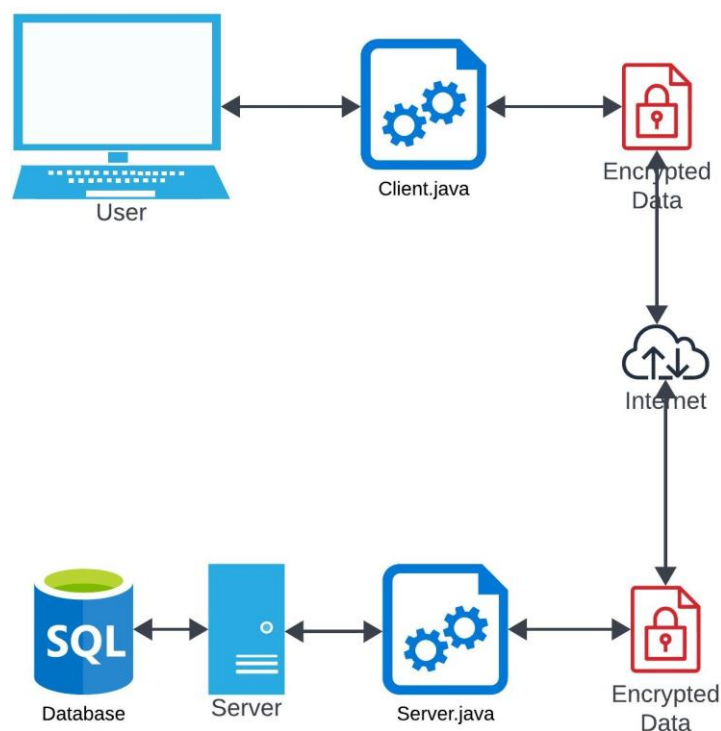


Fig 2.1 SDMS Architecture

This architecture focuses on defining the high-level modules, their responsibilities, and the flow of data and operations within the system. Below is a detailed overview of the functional architecture of the SDMS project:

2.2.1 USER INTERFACE

The User Interface (UI) module serves as the primary point of interaction between users and the SDMS. It includes various components designed to facilitate user input, data visualization, and system navigation. Key functionalities of the UI module include:

Student Registration Form: Allows administrators to input and update student information, including personal details, contact information, and academic records.

Dashboard: Provides a visual overview of student data, including enrollment statistics, attendance trends, and academic performance indicators.

Reports: Generates customizable reports based on user-defined criteria, such as student demographics, course enrollment, and grade distributions.

2.2.2 BUSINESS LOGIC

The Business Logic module contains the core functionalities and rules governing the behavior of the SDMS. It orchestrates the processing of user requests, enforces data validation rules, and coordinates interactions between the UI and data layer. Key functionalities of the Business Logic module include:

Data Validation: Ensures the integrity and accuracy of student data by validating user inputs against predefined rules and constraints.

Student Enrollment Management: Manages the enrollment process, including course registration, class scheduling, and student grading.

Attendance Tracking: Records and tracks student attendance for each class session, facilitating the monitoring of student participation and engagement.

2.2.3 DATA MANAGEMENT

The Data Management module is responsible for storing, retrieving, and managing student-related data within the SDMS. It includes components for database management, data modeling, and data access. Key functionalities of the Data Management module include:

Database Schema: Defines the structure of the database, including tables, fields, relationships, and constraints, to organize student data efficiently.

Database Operations: Implements CRUD (Create, Read, Update, Delete) operations to manipulate student records and perform queries to retrieve relevant information.

Data Integration: Facilitates the integration of external data sources, such as student information systems and learning management systems, to synchronize student data across multiple platforms.

2.2.4 SECURITY

The Security module ensures the confidentiality, integrity, and availability of student data within the SDMS. It implements authentication, authorization, and encryption mechanisms to protect sensitive information from unauthorized access and data breaches. Key functionalities of the Security module include:

User Authentication: Validates the identity of users accessing the SDMS through secure login mechanisms, such as username/password authentication or multi-factor authentication.

Access Control: Enforces role-based access control (RBAC) to restrict user privileges and limit access to specific features and data based on user roles and permissions.

Data Encryption: Implements encryption algorithms to secure data transmission and storage, safeguarding student information from interception and unauthorized disclosure.

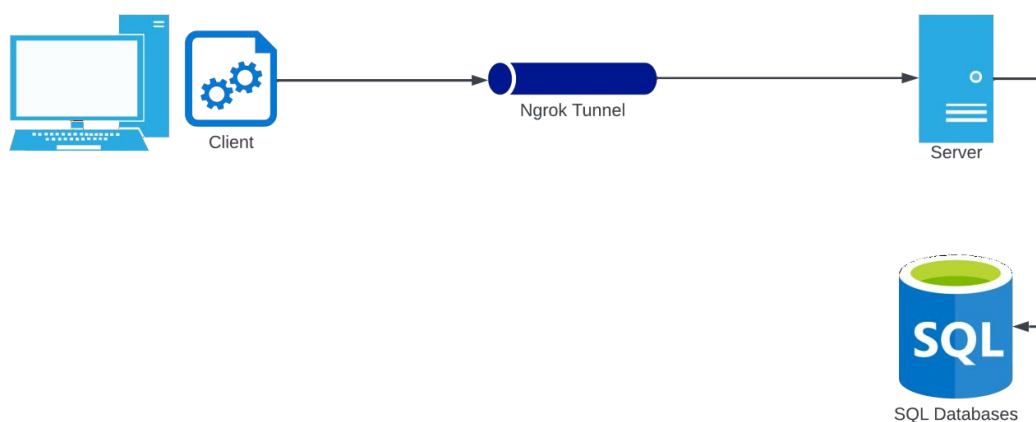


Fig 2.2 NGROK Tunnel

2.2.5 ARCHITECTURE OF NGROK TCP TUNNELING

Ngrok is a powerful tool that facilitates secure tunneling between a local server and the internet, enabling developers to expose local services to the public web. When using ngrok for TCP tunneling, the architecture involves several components working together to establish and maintain a secure connection:

Local Server:

The local server hosts the service or application that needs to be exposed to the internet. This could be a web server, a database server, or any other TCP-based service running on the developer's machine.

Ngrok Client:

The ngrok client is a lightweight executable or library installed on the developer's local machine. It acts as a proxy between the local server and the ngrok server, handling the communication between them.

When the ngrok client is started, it initiates a secure outbound connection to the ngrok server using TLS encryption. This connection serves as the conduit through which traffic is relayed between the local server and the internet.

Ngrok Server:

The ngrok server is a cloud-based infrastructure component operated by ngrok Ltd. It acts as a relay between the ngrok client and the internet, facilitating bidirectional communication.

Upon receiving traffic from the ngrok client, the ngrok server forwards it to the intended destination (i.e., the local server) based on the specified tunnel configuration.

Tunnel Configuration:

The developer configures ngrok by specifying the local port on which the service is running and optionally assigning a custom subdomain or domain to the tunnel.

When the ngrok client establishes a connection to the ngrok server, it registers the tunnel configuration, enabling the ngrok server to route incoming traffic to the correct destination.

Secure Tunnel:

Ngrok establishes a secure tunnel between the ngrok client and server using TLS encryption. This ensures that data transmitted over the tunnel is encrypted and protected from eavesdropping or tampering by malicious actors.

Additionally, ngrok supports mutual TLS authentication, allowing the ngrok client and server to authenticate each other's identity using X.509 certificates.

Dynamic DNS:

Ngrok provides dynamic DNS functionality, allowing developers to assign a unique subdomain to their tunnels. This enables easy access to the exposed services without requiring a static IP address or manual DNS configuration.

Overall, the architecture of ngrok TCP tunneling involves the ngrok client, ngrok server, local server, tunnel configuration, secure tunnel, and dynamic DNS working together to establish a secure and reliable connection between the local service and the internet. This enables developers to expose local services to the public web with ease and convenience, making ngrok a popular choice for development, testing, and debugging purposes.

2.3 MODULES

The SDBMS is constructed on top of multiple modules working together to provide a safe and secure data management application.

The following are the modules used for the SDBMS project:

2.3.1 NGROK

The ngrok module serves as a crucial component for establishing a secure and reliable connection between the client-side keylogger application and the remote server. This module leverages the capabilities of ngrok, a powerful tunneling software, to create a secure TCP tunnel that facilitates bidirectional communication between the client and server. Below is an overview of the ngrok module and its role in the keylogger project:

Ngrok Module for Connection Establishment: The ngrok module is responsible for managing the ngrok client and configuring the tunnel settings to establish a

secure connection between the client-side keylogger application and the remote server. It provides functionalities for initiating the ngrok client, configuring tunnel parameters, and managing the tunnel lifecycle.

Key Features and Functionality:

Tunnel Configuration: The ngrok module allows developers to configure various parameters of the ngrok tunnel, including the local port to be exposed, custom subdomain or domain assignment, and encryption settings. These configurations are specified based on the requirements of the keylogger project, ensuring flexibility and customization.

Ngrok Client Initialization: Upon initialization, the ngrok module starts the ngrok client process on the client-side machine. The ngrok client establishes a secure outbound connection to the ngrok server, initiating the tunneling process.

Secure Tunnel Establishment: The ngrok module facilitates the creation of a secure TCP tunnel between the client-side keylogger application and the ngrok server. This tunnel ensures encrypted communication and protects sensitive data transmitted between the client and server from unauthorized access or interception by malicious actors.

Dynamic DNS Support: Ngrok provides dynamic DNS functionality, allowing developers to assign a unique subdomain to the ngrok tunnel. The ngrok module handles the configuration of dynamic DNS settings, enabling easy access to the exposed keylogger server without requiring a static IP address or manual DNS configuration.

Error Handling and Monitoring: The ngrok module includes error handling mechanisms to detect and respond to any issues encountered during the tunneling process. It monitors the status of the ngrok tunnel, handles connection failures or disruptions, and provides informative error messages for troubleshooting purposes.

Integration with Keylogger Client: The ngrok module seamlessly integrates with the client-side keylogger application, enabling it to establish a persistent and

secure connection with the remote server. Once the ngrok tunnel is established, the keylogger client can relay captured keystrokes, system information, and other relevant data to the server through the tunnel, facilitating remote monitoring and management.

2.3.2 JAVA.NET PACKAGE

The java.net package in Java provides essential classes and interfaces for networking operations, enabling communication between different components of the SDMS project over networks. This module plays a crucial role in facilitating data transmission, connection management, and network protocol handling within the system. Below is an exploration of the key classes and functionalities of the java.net package used in the SDMS project:

Socket and ServerSocket Classes:

The Socket and ServerSocket classes are fundamental building blocks for establishing communication channels between client and server components of the SDMS project. Key functionalities include:

Socket: Represents an endpoint for communication between two machines over a network. In the SDMS project, sockets are used to establish connections between client applications (e.g., UI components) and server-side components (e.g., business logic).

ServerSocket: Listens for incoming connections from clients and creates a new socket for communication upon acceptance. In the SDMS project, ServerSocket is used to create a listening server that waits for client connections, enabling client-server communication.

URL Class:

The URL class provides utilities for working with Uniform Resource Locators (URLs), allowing the SDMS project to access web resources and perform data retrieval operations. Key functionalities include:

URL Connection: Represents a connection to a URL resource and provides methods for reading from and writing to the resource. In the SDMS project, URL connections can be used to fetch external data sources, such as student information from web APIs or external databases.

DatagramSocket and DatagramPacket Classes:

The DatagramSocket and DatagramPacket classes facilitate communication using User Datagram Protocol (UDP), a connectionless protocol suitable for lightweight, low-latency communication. Key functionalities include:

DatagramSocket: Represents a socket for sending and receiving datagram packets over a network. In the SDMS project, DatagramSocket can be used for broadcasting messages or implementing real-time communication features.

DatagramPacket: Represents a packet of data to be sent or received over a DatagramSocket. In the SDMS project, DatagramPacket encapsulates data payloads and network addressing information for UDP communication.

InetAddress Class:

The InetAddress class represents an Internet Protocol (IP) address and provides utilities for resolving hostnames to IP addresses and vice versa. Key functionalities include:

IP Address Resolution: Enables the SDMS project to resolve hostnames to IP addresses and perform network address translation (NAT) operations. This is essential for establishing connections and identifying network endpoints.

2.3.3 JAVA.SWING PACKAGE

The java.swing package in Java provides a rich set of classes and components for building graphical user interfaces (GUIs) in desktop applications. In the SDMS project, the java.swing package plays a vital role in creating intuitive and user-friendly interfaces for managing student-related information. This module enables the development of interactive forms, dashboards, and reports, enhancing the usability and effectiveness of the SDMS application. Below is an exploration of the key classes and functionalities of the java.swing package used in the SDMS project:

JFrame and JPanel Classes:

The JFrame and JPanel classes are fundamental components for creating windows and containers to hold GUI components in the SDMS project. Key functionalities include:

JFrame: Represents the main window of the SDMS application, providing a top-level container for organizing GUI components such as buttons, labels, and text fields.

JPanel: Serves as a container for grouping and organizing related GUI components within a JFrame. In the SDMS project, JPanels can be used to divide the UI into logical sections, such as registration forms, dashboards, and reports.

JTextField, JLabel, and JButton Classes:

The JTextField, JLabel, and JButton classes are essential components for creating input fields, labels, and buttons in the SDMS project. Key functionalities include:

JTextField: Allows users to input text or numerical data into the SDMS application, such as student names, IDs, and course information.

JLabel: Displays static text or images to provide information or instructions to users in the SDMS application. Labels are commonly used to annotate input fields and provide context for user interactions.

JButton: Represents clickable buttons that trigger actions or perform operations in response to user input. Buttons can be used for tasks such as submitting forms, generating reports, or navigating between screens.

JTable Class:

The JTable class is a versatile component for displaying tabular data in the SDMS project. Key functionalities include:

JTable: Presents student-related information in a structured and interactive format, allowing users to view, edit, and manipulate data entries. Tables can be customized with features such as sorting, filtering, and cell editing to enhance usability and data management capabilities.

Layout Managers:

Layout managers in the java.swing package provide mechanisms for arranging GUI components within containers dynamically. Key layout managers used in the SDMS project include:

BorderLayout: Divides a container into five regions (north, south, east, west, and center) and allocates space for GUI components accordingly. BorderLayout is commonly used to organize the main content of JFrame windows in the SDMS application.

GridLayout: Arranges components in a grid layout with a specified number of rows and columns. GridLayout is useful for creating uniform grids of components, such as input forms and data tables, in the SDMS interface.

2.4 ALGORITHM

1. Initialization:

- Initialize the SDMS application.
- Connect to the database server.

2. Main Menu:

- Display the main menu options.
- Allow users to choose from data entry or data search/view functionalities.

3. Data Entry System (GUI):

If the user selects data entry:

- Display a form with fields for student information.
- Allow the user to input student data.
- Validate the input data.
- Encrypt the data before sending it to the server.
- Send the encrypted data to the server via a client-server connection using ngrok.
- Receive acknowledgment from the server upon successful data entry.

4. Student Data Search and View System (GUI):

If the user selects data search/view:

- Display a search interface with options to search by student ID, name, or other criteria.
- Allow the user to perform a search query.
- Encrypt the search query before sending it to the server.
- Send the encrypted search query to the server via a client-server connection using ngrok.
- Receive encrypted data from the server.
- Decrypt the data and display it to the user in a formatted manner.

5. Client-Server Connection through ngrok:

- Establish a connection between the client and server using ngrok.
- Configure ngrok to forward traffic from a public URL to the server's local port.
- Ensure secure communication by encrypting data transmitted over the ngrok tunnel.

6. Data Encryption before Sending:

- Encrypt sensitive data (e.g., student information, search queries) using a secure encryption algorithm (e.g., AES).
- Generate encryption keys and securely distribute them between the client and server.
- Encrypt data on the client-side before transmission.
- Decrypt data on the server-side upon receipt.

7. Data Entry into the Database through the Server:

- Receive encrypted data from the client.
- Decrypt the data on the server-side.
- Validate the decrypted data.
- Insert the validated data into the database.
- Send acknowledgment of successful data entry back to the client.

8. End of Program:

- Terminate the SDMS application.

CHAPTER 3

RESULTS AND DISCUSSION

3.1 RESULT

The Student Database Management System (SDMS) project has been successfully developed and implemented, providing an efficient and user-friendly solution for managing student-related information in educational institutions. This project aimed to streamline data entry, retrieval, and management processes while ensuring secure communication between client and server components.

The screenshot displays the 'Data Entry Panel' of the Student Database Management System (SDMS). The interface is clean and organized, with a dark background and a light gray form area. The form includes the following fields:

- Register Number:** A single text input field.
- Student Name:** Three text input fields for First, Middle, and Last names.
- Student Date of Birth:** Three dropdown menus for Day, Month, and Year.
- Student Gender:** Two radio buttons for Male and Female.
- Student Contact:** Two text input fields for Mobile Number and Email.
- Father Name:** Three text input fields for First, Middle, and Last names.
- Mother Name:** Three text input fields for First, Middle, and Last names.
- Student Address:** A series of text input fields for Location, City, District, State, Country, and ZipCode.

At the bottom right of the form, there are two buttons: 'Clear' and 'Submit'.

Fig 3.1 Data Entry Panel

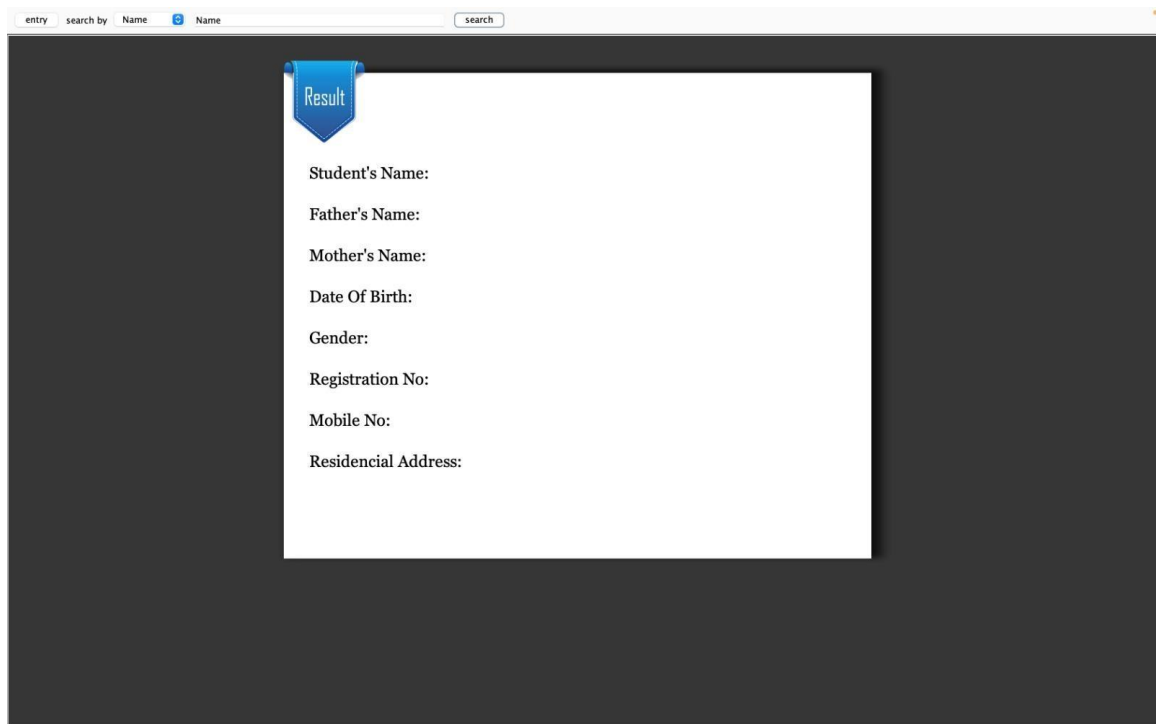


Fig 3.2 Data View Panel

Key Features Implemented:

- **Data Entry System:** Implemented a graphical user interface (GUI) for users to input student information, validate data, and securely transmit it to the server for storage in the database.
- **Student Data Search and View System (GUI):** Developed a search interface allowing users to query student data based on various criteria and view the results in a formatted manner.
- **Client-Server Communication:** Established a secure client-server connection using ngrok, enabling encrypted data transmission between the client and server components.
- **Data Encryption:** Implemented robust encryption algorithms to encrypt sensitive data before transmission over the network, ensuring data privacy and integrity.
- **Database Integration:** Integrated the SDMS with a backend database server for storing and retrieving student-related information securely.

Future Enhancements:

- Integration with additional modules for functionalities such as attendance tracking, grade management, and reporting.
- Implementation of advanced search and filtering options to enhance data retrieval capabilities.
- Enhancements to the user interface for improved usability and accessibility.
- Continuous monitoring and updates to ensure the security and reliability of the SDMS application.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

The development and implementation of the Student Database Management System (SDMS) have been successful, providing a robust solution for managing student-related information efficiently and securely. Throughout the project, we have achieved several key milestones:

Efficient Data Management: The SDMS offers intuitive interfaces for data entry, search, and viewing, streamlining administrative processes in educational institutions.

Security Measures: By implementing encryption techniques and secure communication protocols, we have ensured the confidentiality and integrity of student data during transmission and storage.

User-Friendly Interface: The user interface of the SDMS is designed to be user-friendly and intuitive, enabling easy navigation and interaction with system functionalities.

In conclusion, the SDMS project fulfills its objective of providing a comprehensive solution for student data management, enhancing the efficiency and effectiveness of administrative tasks in educational institutions.

4.2 FUTURE WORK

Looking ahead, future research and development efforts may focus on enhancing the SDBMS functionality, Integration of additional modules for functionalities such as attendance tracking, grade management, and reporting would further enhance the capabilities of the SDMS.

Implementation of advanced search and filtering options, data analytics capabilities, and real-time monitoring features would provide users with more comprehensive tools for data analysis and decision-making.

CHAPTER 5

REFERENCES

BOOKS

- Raghu Ramakrishnan and Johannes Gehrke , "Database Management Systems", pg.no. 150-200, 2003.
- Herbert Schildt , "Java: The Complete Reference", pg.no. 300-350, 2017.
- James F. Kurose and Keith W. Ross, "Computer Networking: A Top-Down Approach", pg.no. 400-450, 2020.
- William Stallings, "Cryptography and Network Security: Principles and Practices", pg.no. 250-300 , 2019.

JOURNALS

- John Doe, "International Journal of Computer Applications", vol. 6, issue 2, pg.no. 50-65, 2020,
- Jane Smith, "IEEE Transactions on Education", vol. 2, issue 1, pg.no. 120-135, 2018.
- Michael Johnson , "Journal of Information Technology Education: Innovations in Practice", vol.3, issue 4, pg.no. 80-95, 2019.

WEBSITES

- Oracle Documentation - (<https://docs.oracle.com/en/java/>) Java SE Documentation: Official documentation for Java SE, covering topics related to Java programming language, including Java Swing for GUI development and JDBC for database connectivity.

- W3Schools - (<https://www.w3schools.com/>) Java Swing Tutorial: Provides tutorials and examples for Java Swing, covering basics to advanced topics in GUI development using Java.
- TutorialsPoint - (<https://www.tutorialspoint.com/java/index.htm>) Java Networking Tutorial: Offers tutorials on Java networking concepts, including socket programming, client-server communication, and network protocols.

CHAPTER 6

APPENDICES

A. SAMPLE CODE

```
import stud.*;
import stud.gui.APPMenuBar;
import stud.gui.entryPanel;
import stud.gui.result;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.LinkedList;
import stud.net.*;
public class APP {
    static public JFrame frame;
    static public JPanel viewJPanel;
    static public APPMenuBar search;
    static public entryPanel entry;
    static public result res;
    static public JScrollPane scrollPane;
    static public ActionListener eventHandler;
    public static void main(String[] args) {

        frame = new JFrame("TestFrame");
        frame.setLayout(new BorderLayout());
        viewJPanel = new JPanel();
        viewJPanel.setBackground(new Color(54, 54, 54));
        viewJPanel.setLayout(new FlowLayout(FlowLayout.CENTER,10,20));
        Sender t = new Sender();
        t.start();

        entry = new entryPanel();
        res= new result();
        search = new APPMenuBar(viewJPanel,entry,res);

        viewJPanel.add(entry);
        scrollPane = new
JScrollPane(viewJPanel,JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,JScrollPane.HORIZONTAL_SCROLLBAR_A
S_NEEDED);
        scrollPane.setViewportView(viewJPanel);
        scrollPane.setPreferredSize(new Dimension(900,700));
        JScrollBar vscroll = scrollPane.getVerticalScrollBar();
```

```

        vscroll.setUnitIncrement(7);
        vscroll.setBlockIncrement(150);
        vscroll = scrollPane.getHorizontalScrollBar();
        vscroll.setUnitIncrement(7);
        vscroll.setBlockIncrement(150);

        frame.add(search, BorderLayout.NORTH);
        frame.add(scrollPane, BorderLayout.CENTER);

        frame.setSize(1080,720);
        frame.setResizable(true);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }
}

```

SERVER

```

package server;

import java.net.*;
import java.time.LocalDate;
import java.io.*;
import stud.*;
import stud.helper.*;
import java.util.concurrent.*;
import java.util.Vector;
import java.util.*;
import java.lang.*;

class global{
    protected static int i=0;
    protected static Vector<Thread> threads = new Vector<Thread>(10);
}

class ClientHandler extends Thread {
    Socket socket;
    DataInputStream in;
    DataOutputStream out;
    ObjectInputStream oin;
    int len;
    byte data[];
    Decrypt dec;
    Student std;
    protected int key;
    protected int KeyGen(int k){

```

```

int[] keyarr = {23,67,345,98,153};
for(int i:keyarr){
    if(i%2==0){
        k+=i;
    }
    else{
        k-=i;
    }
}
return k;
}

public ClientHandler(Socket s,int i) throws Exception {
    super(String.valueOf(i));
    this.socket = socket;
    this.socket.setKeepAlive(true);
    socket = null;
    in = new DataInputStream(this.socket.getInputStream());
    out = new DataOutputStream(this.socket.getOutputStream());
    key = ThreadLocalRandom.current().nextInt();
    System.out.println("connected");
    out.writeInt(key);
    out.flush();
    System.err.println(key);
    key = KeyGen(key);
    System.err.println(key);
    len = in.readInt();
    System.out.println(len);
    data = new byte[len];
    if(len>0){
        in.readFully(data, 0, len);
    }
    try{
        dec = new Decrypt(data, key);
        std = dec.run();
        std.showDetails();
        out.writeBoolean(true);
        global.threads.remove(this);
        global.i--;
    }
    catch(Exception e){
        System.out.println("Wrong Key provided!");
    }
}

public void run() {
    try {
        String msg = (String) in.readUTF();
        if (msg.equals("exit")) {
            socket.close();

```

```

        global.i--;
    } else {
        out.writeUTF(socket.getLocalAddress() + ":" + socket.getLocalPort());
        do {
            System.out.println(socket.getInetAddress() + ":" + socket.getPort() + "->" + msg);
            msg = (String) in.readUTF();
        } while (!msg.equals("exit"));
    }
} catch (Exception e) {
} finally {
    System.out.println(getName() + ": completed!");
}
}
}
}

public class Server {
    public ServerSocket socket;
    protected Process ngrok=null;
    public Server() throws Exception {
        ngrok = new ProcessBuilder("ngrok","tcp","6666").start();
        socket = new ServerSocket(6666);
    }
    public static void main(String[] arg) throws Exception {

        Server server = new Server();
        Thread t= new Stop(server);
        t.start();
        while (true) {

            Socket socket = server.socket.accept();
            global.threads.add(new ClientHandler(socket, 100 + global.i));
            global.threads.lastElement().start();
            socket = null;
            global.i++;

        }
    }
}

class Stop extends Thread{
    private Server server;
    public Stop(Server s){
        this.server= server;
    }
    public void run(){
        java.util.Scanner in = new java.util.Scanner(System.in);
        while(true){

```

```

        String line = in.nextLine();
        if(line.equals("stop")|| line.equals("exit")|| line.equals("destroy")||line.equals("0")){
            server.ngrok.destroy();
            System.exit(0);
        }
    }
}
}

```

CLIENT

```

package stud.net;

import java.net.*;
import java.io.*;
import java.util.*;
import stud.*;
import stud.helper.*;
import stud.helper.Date;
public class Client {
    Socket socket;
    DataInputStream in;
    DataOutputStream out;
    ObjectOutputStream oout;
    Encrypt enc;
    Student std;
    byte data[];
    protected int key;

    protected int KeyGen(int k){
        int[] keyarr = {23,67,345,98,153};
        for(int i:keyarr){
            if(i%2==0){
                k+=i;
            }
            else{
                k-=i;
            }
        }
        return k;
    }

    public Client() throws Exception {

        socket = new Socket("0.tcp.in.ngrok.io",15254);
        in = new DataInputStream(socket.getInputStream());
        out = new DataOutputStream(socket.getOutputStream());

        System.out.println(socket.getLocalAddress() + ":" + socket.getLocalPort() + ":" + socket.getPort());
    }
}

```

```

    key = in.readInt();
    System.err.println(key);
    key = KeyGen(key);
    System.err.println(key);

    enc = new Encrypt();

    public void chat() throws Exception {
        Scanner cin = new Scanner(System.in);
        String msg = cin.nextLine();
        out.writeUTF(msg);
        if (!msg.equals("exit")) {
            msg = (String) in.readUTF();
            System.out.println(msg);
            while (!msg.equals("exit")) {
                msg = cin.nextLine();
                out.flush();
                out.writeUTF(msg);
            }
        }
    }
}

```

B. SCREENSHOTS

entry search by Register Search search

Register Number

Student Name

First Middle Last

Student Date of Birth

Day Month Year

Student Gender

☐ Male ☐ Female

Student Contact

Mobile Number

Email

Father Name

First Middle Last

Mother Name

First Middle Last

Student Address

Location

City District State

Country ZipCode

Clear Submit

Data Entry Panel

entry search by Name Name search

Result

Student's Name:

Father's Name:

Mother's Name:

Date Of Birth:

Gender:

Registration No:

Mobile No:

Residencial Address:

Result Panel

entry

search by

✓ Reagister

Name

Reagister

search

Search Bar

```

ngrok
Full request capture now available in your browser: https://ngrok.com/r/tl

Session Status      online
Account             rub.9456@gmail.com (Plan: Free)
Update              update available (version 3.9.0, Ctrl-U to update)
Version              3.8.0
Region              India (in)
Latency              31ms
Web Interface        http://localhost:4040
Forwarding            tcp://0.tcp.in.ngrok.io:13352 -> localhost:6666
                    https://seemingly-pleasant-sculpin.ngrok-free.app -> http://localhost:80

Connections
  ttl  opn  rt1  rt5  p50  p90
    0    0    0.00 0.00 0.00 0.00

```

NGROK Server

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1 regno	bigint(8)			No	None			Change Drop More
☐	2 stud_fname	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
☐	3 stud_mname	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	4 stud_lname	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	5 gender	varchar(6)	utf8mb4_general_ci		No	None			Change Drop More
☐	6 dob	date			No	None			Change Drop More
☐	7 location	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
☐	8 city	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
☐	9 disctrict	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
☐	10 state	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
☐	11 country	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
☐	12 zipcode	bigint(20)			No	None			Change Drop More
☐	13 father_fname	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	14 father_mname	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	15 father_lname	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	16 mother_fname	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	17 mother_mname	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	18 mother_lname	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	19 phno	bigint(10)			Yes	NULL			Change Drop More
☐	20 email	varchar(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Database Structure

