STUDY PLANNER USING JAVA

MINOR PROJECT REPORT

By

AEKEESH JAISWAL(RA2311003030006)
VRINDA AGGARWAL(RA2311003030017)
SHAMBHAVI TRIVEDI(RA2311003030051)
VAISHNAVI SINGH(RA2311003030064)

Under the guidance of

MS. SHRUTHY GOVINDAN

In partial fulfilment for the Course

of

21CSC203P – ADVANCED PROGRAMMING PRACTICE

in COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

DELHI-NCR CAMPUS

NOVEMBER-2024

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC203P ADVANCED PROGRAMMING PRACTICE** entitled in "**STUDY PLANNER USING JAVA**" is the bonafide work of **AEKEESH JAISWAL(RA2311003030006), VRINDA AGGARWAL (RA2311003030017), SHAMBHAVI TRIVEDI (RA2311003030051),VAISHNAVI SINGH(RA2311003030064)** who carried out the work under my Supervision.

SIGNATURE

Ms. Shruthy Govindan

Assistant Professor

Computer Science and Engineering

SRM Institute of Science and Technology

Delhi-Ncr Campus

ACKNOWLEDGEMENT

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. R.P. Mahapatra** for bringing out novelty in all executions.

We are highly thankful to our my Course project Faculty **MS. SHRUTHY GOVINDAN , ASSISTANT PROFESSOR , CSE,** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. Avneesh Vashistha, HOD(ASSOCIATE PROFESSOR) , CSE** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

ABSTRACT

The study planner application developed using Java aims to enhance the academic success of students by providing a structured and efficient tool for managing study schedules, tasks, and goals. In today's fast-paced educational environment, students often struggle with time management, leading to increased stress and diminished academic performance. This application addresses these challenges by offering a user-friendly interface that enables students to create, organize, and prioritize their study tasks.

Key features of the study planner include task management, which allows users to set deadlines, categorize tasks by subject, and track progress through visual metrics. Additionally, the application incorporates reminder notifications to help students stay on track with their study schedules. The integration of motivational quotes fosters a positive mindset and encourages consistent study habits.

The application is built using Java, leveraging its robust libraries and frameworks for both desktop and web versions. Data security is prioritized through secure user authentication, encryption, and regular backup mechanisms, ensuring user data remains safe and accessible. The study planner not only facilitates better organization and productivity but also promotes a balanced approach to academic life.

TABLE OF CONTENTS

1. INTRODUCTION

This project explores the creation of a study planner application using Java. The planner will offer a user-friendly interface to manage and organize study schedules. It will provide key features such as task creation and deadline setting.The project will utilize Java's object-oriented programming paradigm to encapsulate data and functionality.

The planner will be designed with a focus on flexibility and adaptability. It will cater to various learning styles and study habits. The application will aim to enhance students' time management skills, reduce stress, and improve overall academic performance. Java's robustness and versatility make it an ideal language for building a reliable and efficient study planner.

1.1 MOTIVATION

1.The study planner aims to improve students' time management skills by providing a structured framework for organizing their academic workload.

2. By breaking down tasks into manageable chunks and setting realistic deadlines, students can gain a sense of control over their studies. This can reduce stress and anxiety, leading to a more relaxed and focused learning environment.

3. Increased productivity is another key benefit. With a clear plan and progress tracking features, students can maintain momentum and stay motivated throughout their studies.

4.Regular reminders and personalized settings further enhance focus and efficiency.

1.2 OBJECTIVE

1. Enhance Organization

- **Objective**: To provide a structured layout for scheduling study sessions, assignments, and deadlines, making it easier to manage academic responsibilities.

2. Improve Time Management

- **Objective**: To help students allocate their time effectively, balancing study sessions with other commitments and ensuring all tasks are completed on time.

3. Set Clear Goals

- **Objective**: To establish specific, measurable, achievable, relevant, and time-bound (SMART) goals for both short-term and long-term academic achievements.

4. Increase Focus and Concentration

- **Objective**: To create designated study blocks that minimize distractions and promote deeper engagement with the material.

5. Facilitate Progress Tracking

- **Objective**: To enable students to monitor their completion of tasks, assess their productivity, and make adjustments as necessary to stay on track.

1.3 PROBLEM STATEMENT

Problem Statement for Study Planner in Java:

Many students struggle with managing their time and tasks effectively, leading to increased stress, poor academic performance, and feelings of overwhelm. This issue is exacerbated by competing responsibilities, such as extracurricular activities, part-time jobs, and personal commitments, which can make it challenging to prioritize study time and meet deadlines. The study planner aims to provide a comprehensive tool that helps students organize their study schedules, manage their time effectively, set clear goals, and track their progress.

2. LITERATURE SURVEY

A literature survey on study planners developed using Java involves examining existing research, projects, and applications related to the concept. Here's a structured overview of key themes and findings in this area:

1. Introduction to Study Planners

- **Purpose**: Study planners are tools designed to help students manage their study schedules, prioritize tasks, and track progress. They can significantly improve time management and reduce stress.
- **Technological Integration**: Java, as a versatile programming language, is often used for developing educational tools due to its platform independence and robust libraries.

2. Existing Study Planner Applications

- **Desktop Applications**: Several applications leverage Java Swing or JavaFX to create user-friendly interfaces for study planners, enabling features such as task management, reminders, and progress tracking.
    - **Example**: Applications like "JStudy" use Java to offer a simple interface for creating and managing study schedules.
- **Web-Based Solutions**: Some planners are built using Java-based web frameworks (e.g., Spring, JavaServer Faces), allowing for real-time collaboration and access from various devices.
    - **Example**: Online study planners like "MyStudyLife" incorporate Java backends to manage user data securely.

3. Features and Functionalities

- **Task Management**: Most planners allow users to create, edit, and delete tasks, set deadlines, and categorize tasks by subject or priority.
- **Progress Tracking**: Many implementations include features for tracking the completion of tasks, often using visual indicators or graphs.
- **Reminders and Notifications**: Java's integration with APIs enables planners to send reminders via email or mobile notifications.
- **Motivational Elements**: Some planners incorporate motivational quotes or gamification elements to encourage consistent study habits.

4. User Interface Design

- **Usability Studies**: Research has shown that user-friendly interfaces significantly impact the effectiveness of study planners. Java's GUI libraries (like Swing and JavaFX) provide tools for creating intuitive interfaces.
- **Accessibility**: Effective study planners consider accessibility features to cater to all users, ensuring that students with disabilities can utilize the planner effectively.

5. Impact on Learning Outcomes

- **Academic Performance**: Studies indicate that students using structured study planners often achieve better academic outcomes compared to those who do not.
- **Stress Reduction**: By providing a clear overview of tasks and deadlines, planners help mitigate anxiety and improve overall mental well-being.

6. Challenges and Limitations

- **User Engagement**: Maintaining user engagement can be challenging, as students may abandon planners if they perceive them as cumbersome or unnecessary.
- **Technological Barriers**: Some students may face difficulties in navigating digital tools, highlighting the need for comprehensive user support.

7. Future Directions

- **Integration with Learning Management Systems (LMS)**: Future planners could integrate with popular LMS platforms to automatically import deadlines and assignments.
- **Data Analytics**: Utilizing Java's capabilities for data analytics could allow planners to offer personalized study recommendations based on user performance and habits.
- **Mobile Applications**: With the growing use of smartphones, developing Java-based mobile applications could increase accessibility and usability for students on the go.

3.REQUIREMENTS

2.1 Requirement Analysis

Requirements for a Study Planner in Java:

1. **User Authentication**
    - **Users should be able to create accounts and log in securely.**
    - **Password recovery options should be available.**

2. **Task Management**

    - **Users should be able to create, edit, and delete tasks.**
    - **Each task should include attributes such as title, description, deadline, priority level, and status (e.g., pending, completed).**

3. **Scheduling**

    - **Users should be able to set specific study times for each task.**
    - **The planner should allow users to view tasks in different formats (e.g., daily, weekly, monthly).**

4. User Interface

- **A clear, user-friendly interface should be designed, with intuitive navigation.**
- **The interface should be responsive, ensuring usability on various screen sizes if applicable.**

2.2 Software Requirement

When developing a study planner application, defining the software requirements is crucial for ensuring functionality, usability, and compatibility. Here's a comprehensive list of software requirements:

- **WINDOWS OS**
- **JAVA DEVELOPMENT KIT (JDK): version JDK 8 or higher**
- **IDE : BlueJ**
- **JDBC Conectivity issued by NeonTech host server**
- **PostgreSQL Database**
- **PostgreSQL JDBC Driver: postgresql-42.x.x.jar.**

2.3 Hardware Requirement

The hardware requirements for a Study Planner in Java can vary based on the scale of the system, the number of users, and the expected workload. However, for a typical small to medium-sized bug tracking system, the following hardware requirements should suffice:

1. **Server**:

- **RAM: At least 4GB RAM (8GB recommended for smooth performance)**
- **STORAGE: 500 MB or more for JDK, JDB Driver, IDE, and PostgreSQL files.**

2. **Network**:

- **Reliable internet connection for remote database access.**

2.4 DATA REQUIREMENTS

When developing a study planner application, clearly defining the data requirements is essential for ensuring effective data management and functionality. Here's a comprehensive overview of the data requirements for such an application:

1. User Data

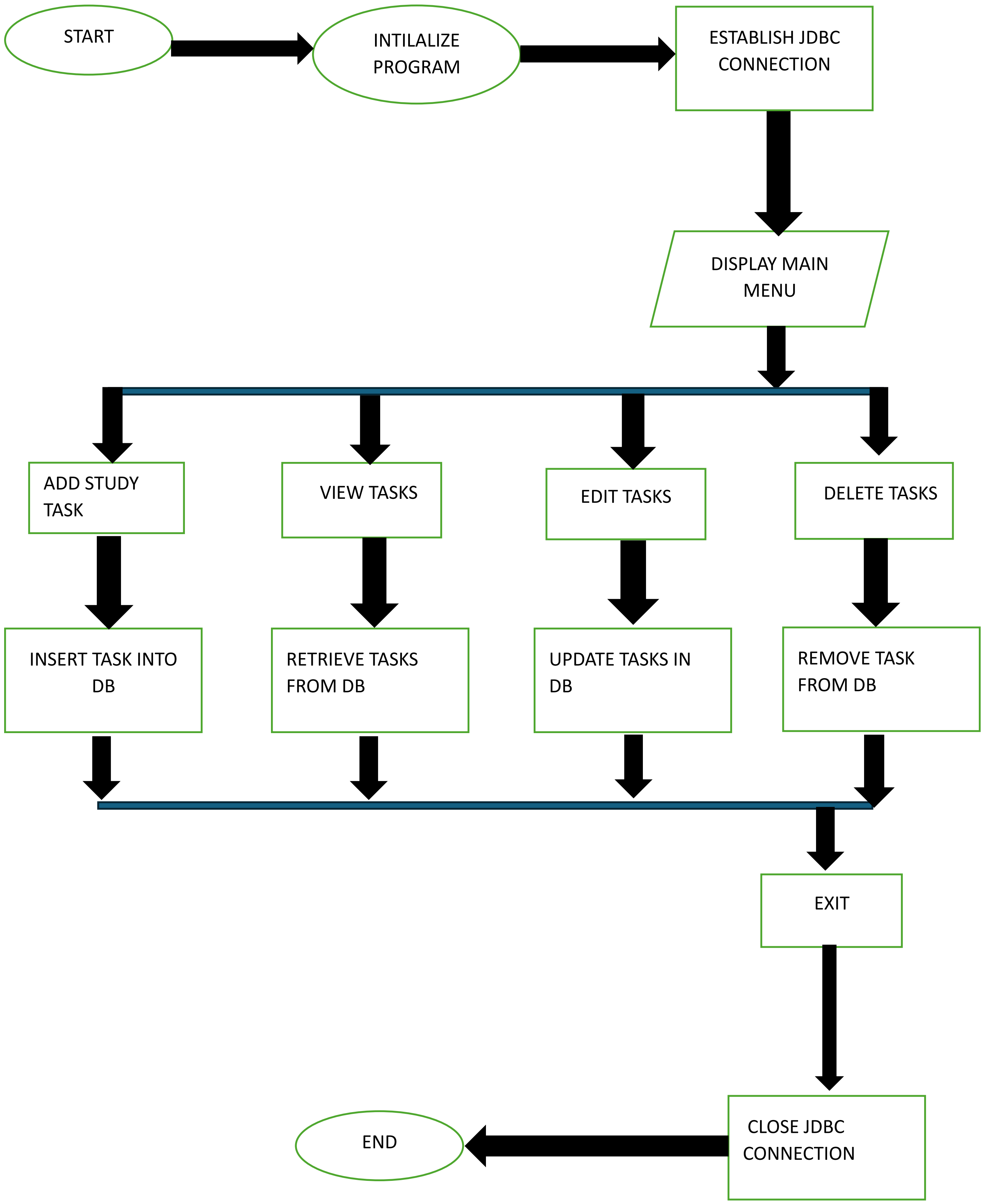
- **User Profile:**
  - **User ID: Unique identifier for each user.**
  - **Username: User's chosen display name.**
  - **Email Address: For account verification and notifications.**
  - **Password: Securely hashed for authentication.**
  - **Profile Picture: Optional, for personalization.**
  - **Preferences: User-specific settings (e.g., theme, notification preferences).**

2. Task Management Data

- **Tasks:**
  - **Task ID: Unique identifier for each task.**
  - **User ID: Identifier linking the task to a specific user.**
  - **Title: Brief description of the task.**
  - **Description: Detailed information about the task (optional).**
  - **Due Date: Date and time the task is due.**
  - **Priority Level: Importance of the task (e.g., high, medium, low).**
  - **Status: Current state of the task (e.g., pending, completed).**
  - **Category: Subject or category to which the task belongs (e.g., Math, History).**

4.ARCHITECTURE AND DESIGN

STUDY PLANNER FLOW CHART



STUDY PLANNER UML DIAGRAMS

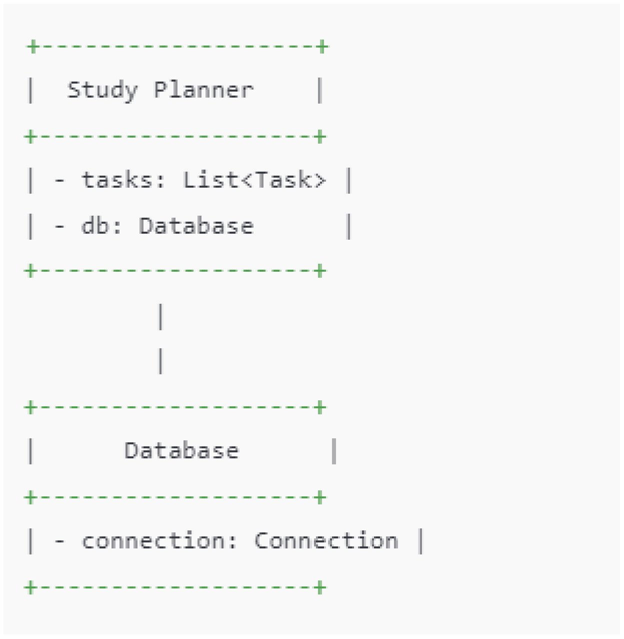
1.CLASS DIAGRAM

```
+--------------------+
 | MainFrame |
+--------------------+
| - loginPanel |
| - studyPlanPanel |
 +--------------------+
| + showLogin() |
| + showStudyPlan() |
+--------------------+
+--------------------+

| LoginPane l |
 +--------------------+
| - usernameField |
| - passwordField |
+--------------------+
| + validateUser() |
+--------------------+

+--------------------+
| StudyPlanPanel |
+--------------------+
| - studyPlanList |
| - taskPanel       |
+--------------------+
| + displayPlans() |
| + selectPlan()  |
+--------------------+
 | TaskPanel        |
 +--------------------+
 | - taskList         |
+--------------------+
| + addTask()       |
| + removeTask() |
 | + updateTask() |
+--------------------+

+--------------------+
| User                   |
+--------------------+
| - username       |
| - password        |
+--------------------+

+--------------------+
| StudyPlan         |
+--------------------+
| - planName      |
| - tasks: List<Task>|
+--------------------+

+--------------------+
| Task                   |
+--------------------+
| - title                |
| - description    |
| - deadline         |
| - status             |
+--------------------+
+--------------------+
| StudyPlanController |
+--------------------+
| + createPlan()   |
| + addTask()      |
| + removeTask() |
| + updateTask() |
| + viewPlan()     |
+--------------------+
```

2.SEQUENCE DIAGRAM

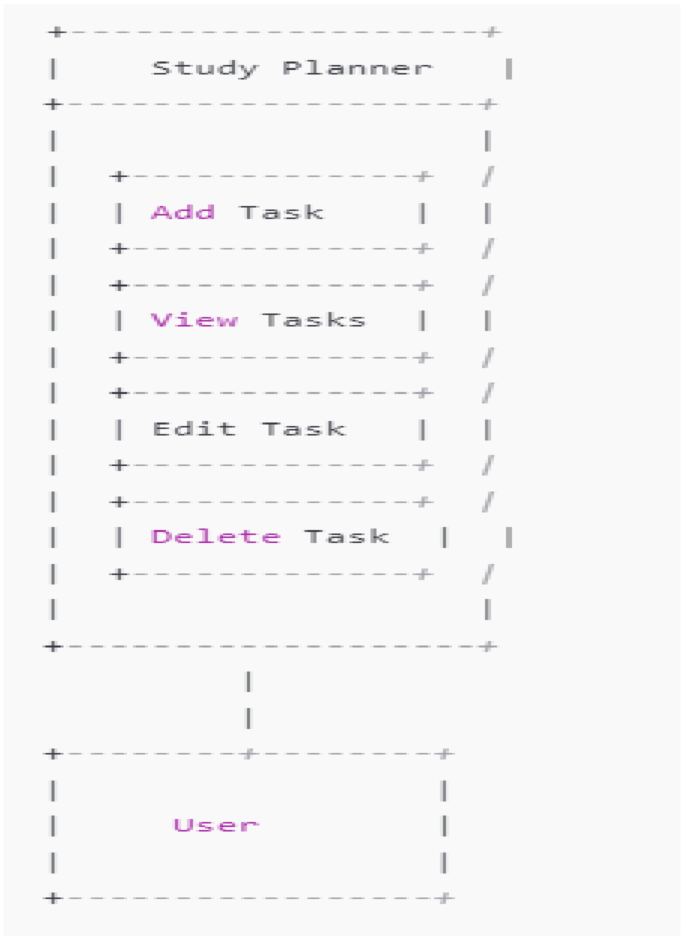
```
+--------------------+          +--------------------+          +--------------------+
|    User            |          |  StudyPlanner      |          |   Database         |
+--------------------+          +--------------------+          +--------------------+
        |                                  |                                  |
        |        displayMenu()            |                                  |
        |--------------------------------->|                                  |
        |                                  |                                  |
        |         addTask()               |                                  |
        |--------------------------------->|                                  |
        |                                  |                                  |
        |         inputTaskDetails()      |                                  |
        |--------------------------------->|                                  |
        |                                  |                                  |
        |         connectDB()             |                                  |
        |                                  |----------> establishConnection()  |
        |                                  |                                  |
        |                                  |<---------- connection established   |
        |                                  |                                  |
        |         insertTask(Task)        |                                  |
        |                                  |----------> insertTask(Task)       |
        |                                  |                                  |
        |                                  |<---------- task inserted          |
        |         closeDB()               |                                  |
        |                                  |----------> closeConnection()      |
        |                                  |                                  |
```

3.COMPONENT DIAGRAM



```
+-------------------+
|  Study Planner    |
+-------------------+
| - tasks: List<Task> |
| - db: Database      |
+-------------------+
         |
         |
+-------------------+
|      Database      |
+-------------------+
| - connection: Connection |
+-------------------+
```

4.USE CASE DIAGRAM



```
+----------------------+
|     Study Planner     |
+----------------------+
|                      |
|   +-------------+    /
|   | Add Task    |    |
|   +-------------+    /
|   +-------------+    /
|   | View Tasks  |    |
|   +-------------+    /
|   +-------------+    /
|   | Edit Task   |    |
|   +-------------+    /
|   +-------------+    /
|   | Delete Task  |    |
|   +-------------+    /
|                      |
+----------------------+
          |
          |
+---------+---------+
|                   |
|      User         |
|                   |
+-------------------+
```

5.IMPLEMENTATION

● PROGRAM FOR STUDY PLANNER :

```
import java.sql.*;

import java.util.Scanner;

 class Task {

 private String subject;

private String description;

private String deadline;

private int priority;

  public Task(String subject, String description, String deadline, int priority) {

       this.subject = subject;

       this.description = description;

       this.deadline = deadline;

       this.priority = priority;

   }

   public String getSubject() {
        return subject;

 }

 public String getDescription() {

      return description;

    }

public String getDeadline() {

        return deadline;

    }

   public int getPriority() {

        return priority;

    }

    @Override

    public String toString() {

        return "Subject: " + subject + "\nDescription: " + description + "\nDeadline: " + deadline +
 "\nPriority: " + priority;

    }

 }
```

```
class StudyPlanner {

    private Connection connection;

    public StudyPlanner() throws SQLException {

        try {

            // Update connection string if needed

            connection = DriverManager.getConnection(

                "jdbc:postgresql://ep-red-recipe-a5ip2btu.us-east-
2.aws.neon.tech/neondb?sslmode=require",

                "neondb_owner",

                "AVNMtfBZh37L"

            );

            System.out.println("Database connection successful!");

        } catch (SQLException e) {

            System.out.println("Database connection failed!");

            e.printStackTrace();

            throw e;

            }

    }
    public void addTask(Task task) {

        String query = "INSERT INTO tasks (subject, description, deadline, priority) VALUES (?, ?, ?,
?)";

        try (PreparedStatement stmt = connection.prepareStatement(query)) {

            stmt.setString(1, task.getSubject());

            stmt.setString(2, task.getDescription());

            stmt.setDate(3, Date.valueOf(task.getDeadline()));

            stmt.setInt(4, task.getPriority());

            stmt.executeUpdate();

            System.out.println("Task added to the database.");

        } catch (SQLException e) {

            System.out.println("Error adding task to the database.");

            e.printStackTrace();

        }

    }

    public void viewTasks() {

        String query = "SELECT * FROM tasks";

        try (Statement stmt = connection.createStatement();
     ResultSet rs = stmt.executeQuery(query)) {

         if (!rs.isBeforeFirst()) {

        System.out.println("No tasks available.");

        return;

    }

 while (rs.next()) {

        System.out.println("ID: " + rs.getInt("id"));

        System.out.println("Subject: " + rs.getString("subject"));

        System.out.println("Description: " + rs.getString("description"));

        System.out.println("Deadline: " + rs.getDate("deadline"));

        System.out.println("Priority: " + rs.getInt("priority"));

        System.out.println();

    }

} catch (SQLException e) {

    System.out.println("Error retrieving tasks.");

    e.printStackTrace();

}
```

```
}

   public void removeTask(int id) {

    String query = "DELETE FROM tasks WHERE id = ?";

    try (PreparedStatement stmt = connection.prepareStatement(query)) {

        stmt.setInt(1, id);

        int rowsAffected = stmt.executeUpdate();

        if (rowsAffected > 0) {

            System.out.println("Task removed from the database.");

        } else {

            System.out.println("Task not found.");

        }

    } catch (SQLException e) {

        System.out.println("Error removing task from the database.");

        e.printStackTrace();

    }

}
```

```
    public void closeConnection() {
        try {
            if (connection != null && !connection.isClosed()) {
                connection.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        StudyPlanner planner;


        try {
            planner = new StudyPlanner();
```

```
  } catch (SQLException e) {
      System.out.println("Unable to initialize planner. Exiting.");
      return;
  }
while (true) {
      System.out.println("\n--- Study Planner ---");
      System.out.println("1. Add Task");
      System.out.println("2. View Tasks");
      System.out.println("3. Remove Task");
      System.out.println("4. Exit");
      System.out.print("Choose an option: ");
      int option = scanner.nextInt();
      scanner.nextLine(); // Consume newline
switch (option) {
        case 1:
            System.out.print("Enter subject: ");
            String subject = scanner.nextLine();
```

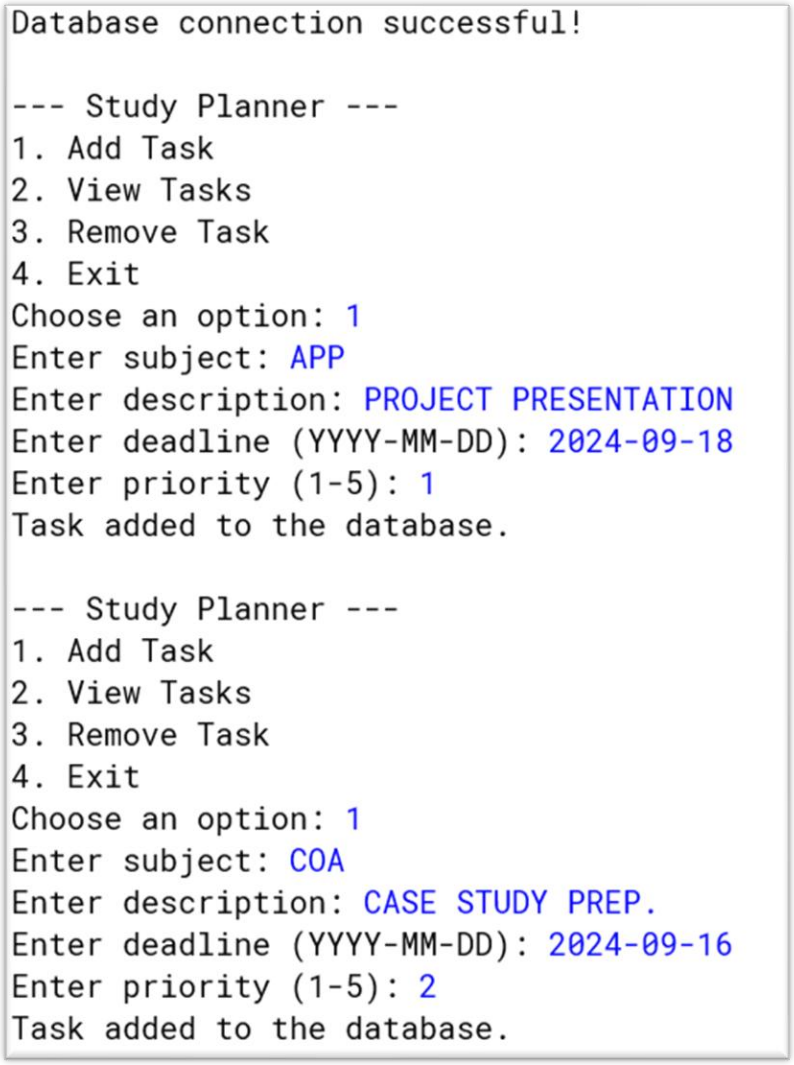
```
    System.out.print("Enter description: ");

    String description = scanner.nextLine();

    System.out.print("Enter deadline (YYYY-MM-DD): ");

    String deadline = scanner.nextLine();

    System.out.print("Enter priority (1-5): ");

    int priority = scanner.nextInt();

    scanner.nextLine(); // Consume newline

   Task task = new Task(subject, description, deadline, priority);

    planner.addTask(task);

    break;

case 2:

    planner.viewTasks();

    break;

case 3:

    System.out.print("Enter the task ID to remove: ");

    int id = scanner.nextInt();

    planner.removeTask(id);
```

```
                    break;

                case 4:

                    planner.closeConnection();

                    System.out.println("Exiting...");

                    scanner.close();

                    return;

                default:

                    System.out.println("Invalid option.");

            }

        }

    }

}
```

6. RESULTS AND DISCUSSION

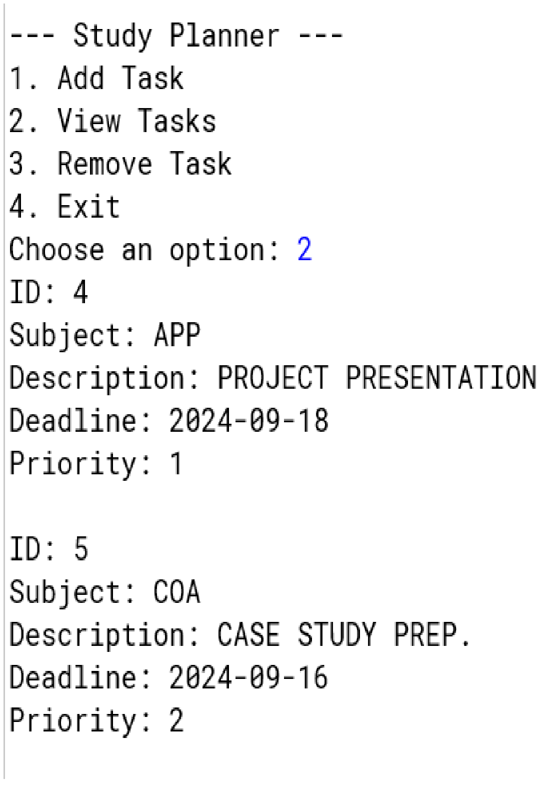
OUTPUT :

ADDING TASKS



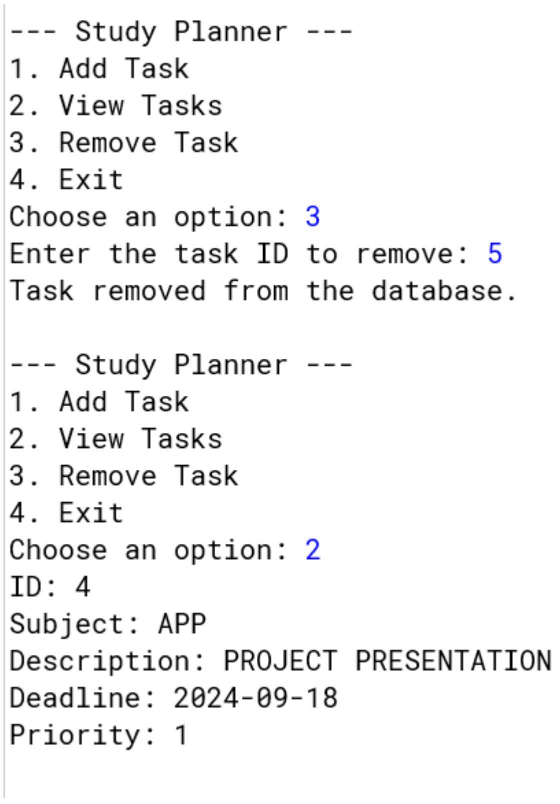
```
Database connection successful!

--- Study Planner ---
1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Choose an option: 1
Enter subject: APP
Enter description: PROJECT PRESENTATION
Enter deadline (YYYY-MM-DD): 2024-09-18
Enter priority (1-5): 1
Task added to the database.

--- Study Planner ---
1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Choose an option: 1
Enter subject: COA
Enter description: CASE STUDY PREP.
Enter deadline (YYYY-MM-DD): 2024-09-16
Enter priority (1-5): 2
Task added to the database.
```

VIEWING TASKS



```
--- Study Planner ---
1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Choose an option: 2
ID: 4
Subject: APP
Description: PROJECT PRESENTATION
Deadline: 2024-09-18
Priority: 1

ID: 5
Subject: COA
Description: CASE STUDY PREP.
Deadline: 2024-09-16
Priority: 2
```

REMOVING TASKS



```
--- Study Planner ---
1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Choose an option: 3
Enter the task ID to remove: 5
Task removed from the database.

--- Study Planner ---
1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Choose an option: 2
ID: 4
Subject: APP
Description: PROJECT PRESENTATION
Deadline: 2024-09-18
Priority: 1
```

7 . CONCLUSION

In conclusion, A study planner built with Java and JDBC connectivity enables users to effectively manage their study schedules by interacting with a database for data storage. This setup ensures efficient task management through features like adding, updating, and viewing study tasks, while also allowing for data persistence. By combining Java's programming capabilities with JDBC's database functionality, you create a practical and user-friendly tool that helps users stay organized in their academic pursuits.

Quality Assurance: A Bug Tracking System is paramount for maintaining and improving the quality of software products. It enables teams to identify and address defects, ensuring that the final product is robust and reliable.

Efficiency: Features such as drag-and-drop scheduling, quick task entry, and customizable views (daily, weekly, monthly) allow users to organize their study sessions intuitively. Automated reminders and notifications help keep students on track, minimizing the risk of missed deadlines and maximizing productive study time.

Transparency: achieved through clear visual metrics and reporting features. Users can easily track their progress, view completed tasks, and analyze time spent on different subjects. This visibility helps students understand their study patterns, making it easier to identify areas for improvement and adjust their schedules accordingly.

Customization The application supports extensive customization options, enabling users to tailor their experience to their preferences. Students can categorize tasks by subject, set priority levels, and personalize reminder settings. The option to add motivational quotes and customize themes fosters a more engaging and personalized study environment.

Integration The study planner can integrate with other educational tools and platforms, such as Learning Management Systems (LMS) or calendar applications. This integration allows users to sync assignments and deadlines automatically, creating a cohesive study ecosystem that simplifies task management and enhances overall productivity.

Security : it is a top priority, with measures including user authentication, data encryption, and regular backups. The application ensures that sensitive user information is protected from unauthorized access and data breaches.

User-Friendly Interface : The user-friendly interface is designed with simplicity and accessibility in mind. Intuitive navigation, clear labels, and visual elements ensure that users can easily find and utilize features without extensive training.

Overall, the development of a study planner application using Java represents a significant step toward addressing the challenges students face in managing their academic responsibilities. By providing a structured platform for organizing tasks, setting deadlines, and tracking progress, this application empowers users to take control of their study schedules. The integration of features such as reminders, motivational quotes, and detailed analytics further enhances the user experience, fostering better study habits and improved academic performance.

8. REFERENCES

1. STUDY PLANNER (Project Ideas):
   https://codegym.cc/groups/posts/412-how-to-create-an-effective-study-plan-8-steps-for-java-learners

2. Chat GPT : https://chat.openai.com

3. Javatpoint : https://javatpoint.com