DIGITAL ASSIST ORGANIZER

1.Company Profile - GUVI HCL

GUVI HCL is a collaborative initiative between GUVI (Grab Ur Vernacular Imprint) and HCL Technologies, designed to bridge the gap between academic learning and industry skills through hands-on technical training and real-world exposure.

GUVI, an edtech platform incubated by IIT Madras and IIM Ahmedabad, was founded in 2014 with the mission of making technology education accessible to everyone in vernacular languages such as Tamil, Telugu, Hindi, and Kannada. Headquartered in Chennai, India, GUVI has empowered over 10 lakh learners through online courses, coding bootcamps, and career programs. The platform specializes in programming, full-stack development, artificial intelligence, cloud computing, and data science, offering training aligned with current IT industry needs.

HCL Technologies, a global technology leader with decades of experience in IT services and consulting, partners with GUVI to offer industry-relevant programs like the GUVI-HCL Tech Career Program and HCL Career Launchpad. Through this collaboration, students gain exposure to enterprise-level technologies, mentorship from HCL professionals, and opportunities to work on real-time industrial projects.

The GUVI-HCL partnership focuses on transforming aspiring students into skilled and job-ready IT professionals by integrating theoretical learning with practical implementation. Together, they aim to create a new generation of tech talent that is proficient, confident, and ready to contribute to India's fast-growing digital and innovation ecosystem.

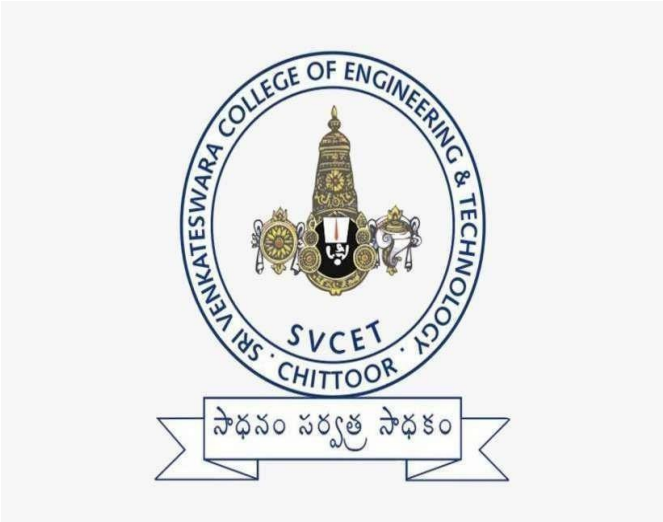
SRI VENKATESWARA COLLEGE 0F ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

R.V.S.Nagar,Chittoor–517127.(A.P)

(ApprovedbyAICTE, NewDelhi,AffiliatedtoJNTUA,Anantapur)

(Accredited by NBA, New Delhi C NAAC, Bangalore)
(AnISO9001:2000Certified Institution)
2025-2026



This is to certify that the"Project report"submitted by **PURINI LAVANYA** (Regd.No.:22781A05C0) is workdone by her and submitted during 2025-2026.Academic year, in partial fulfilment of the requirements for the award of the Degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE ENGINEERING**, at The **HCL GUVI**.

Mr. P. RAGAVAN
GUVI-HCL
(Technical Trainer)

HOD NAME: Dr.P.JYOTHEESWARI
Head of the Department
(Department Of CSE)

INDEX

1. Abstract

The Digital Assist Organizer is an intelligent productivity tool designed to streamline personal and professional task management through smart scheduling, automated reminders, seamless integration with digital calendars and communication platforms. Leveraging AI-driven insights, it prioritizes tasks, adapts to user behavior, and optimizes time allocation improved efficiency and reduced cognitive load.

The system features a user-friendly interface that supports voice commands, natural language input, and cross-device synchronization, making it accessible and effective for a wide range of users. By combining automation with personalization, Digital Assist Organizer transforms traditional to-do lists into dynamic, responsive productivity systems tailored to individual needs.

The **Digital Assist Organizer** is a Java-based productivity application designed to help users — including students, developers, and professionals — effectively manage tasks, schedules, and deadlines. Developed using **JavaFX** for a modern and interactive user interface, and integrated with a **MySQL** database for secure data storage, the application provides features such as task creation, calendar scheduling, automated reminders, and priority-based task organization.

This cross-platform solution leverages the power of **object-oriented programming modular design** to ensure maintainability and scalability. Users access personalized daily, weekly, monthly overviews, set notifications for important events, and streamline their workflows. Whether used for academic planning, software development tracking, professional scheduling, the Digital Assist Organizer offers a unified and efficient approach to personal organization. Its user-centric design and extensible architecture make it a versatile tool adaptable to various productivity needs.

The project emphasizes software engineering principles, including modular design, object-oriented programming, and structured data management. It demonstrates automation in employee management improve efficiency and transparency within an organization.

2. Introduction

In today's fast-paced digital world, effective time and task management have become essential for productivity and success. Students must balance academic responsibilities, assignments, and extracurricular activities; developers juggle project deadlines, code reviews, and learning goals; and professionals manage meetings, reports, and strategic planning. Keeping track of these tasks across different platforms and tools can be overwhelming and inefficient.

Digital Assist Organizer is a comprehensive Java-based application designed to simplify and centralize task management for a wide range of users. With features such as intelligent task scheduling, priority-based organization, calendar integration, automated reminders, tool helps users stay organized, focused, and control of their daily routines. Built using **JavaFX** for a responsive graphical interface and **MySQL** for data storage, the system ensures both usability and reliability.

This application bridges the gap between basic to-do lists and complex project management tools offering lightweight, customizable, and intuitive solution. Whether it's for managing coursework, development milestones, or professional commitments, the Digital Assist Organizer provides an efficient digital companion to boost productivity and reduce mental load.

In an era where digital demands are constantly increasing, managing time and tasks efficiently has become crucial challenge for individuals across all fields. Whether it's students keeping track of assignments and exam schedules, developers organizing coding tasks and project deadlines, or professionals balancing meetings, reports, and personal goals—staying organized is key to productivity and success.

This project showcases the application of Java programming concepts such as encapsulation, inheritance, and collections, combined with database connectivity through JDBC (Java Database Connectivity).

3. Objectives

The main objective of the **Digital Assist Organizer** is to develop a cross-functional, Java-based application that enhances productivity and time management for students, developers, and professionals. The key objectives include:

1. **Task Management**
   To provide users with the ability to create, update, categorize, and delete tasks efficiently through a user-friendly interface.
2. **Scheduling and Calendar Integration**
   To implement a calendar view that helps users visualize their tasks and deadlines on a daily, weekly, or monthly basis.
3. **Priority and Deadline Tracking**
   To allow users to set task priorities and deadlines, enabling better time allocation and decision-making.
4. **Automated Reminders and Notifications**
   To integrate a reminder system that alerts users of upcoming tasks or overdue deadlines.
5. **User-Friendly Interface**
   To design a clean and intuitive GUI using **JavaFX**, ensuring accessibility for users with varying technical skills.
6. **Persistent Data Storage**
   To use **MySQL** for securely storing user data, ensuring tasks are saved and retrievable across sessions.
7. **Cross-User Applicability**
   To design the system in a way that caters to different user roles — students (academic tasks), developers (project milestones), and professionals (meeting schedules, work tasks).
8. **Scalability and Extensibility**
   To build the system using modular, object-oriented design principles to support future upgrades and additional features (e.g., cloud sync, mobile support).

4. Algorithm and Workflow

Algorithm Steps:

Algorithm: Digital Assist Organizer

1. **Start**
2. Initialize application components (GUI, database connection, user session)
3. Display **Home Dashboard** (tasks summary, calendar view)
4. Wait for user input:
    - **Create Task**
    - **Edit/Delete Task**
    - **View Calendar**
    - **Set Reminder**
    - **Check Notifications**
    - **Exit Application**
5. If user selects **Create Task**:
    - Prompt for task details (title, description, due date, priority, category)
    - Validate input
    - Save task to MySQL database
    - Update GUI task list
6. If user selects **Edit/Delete Task**:
    - Fetch task from database
    - Perform update or deletion
    - Refresh task view
7. If user selects **View Calendar**:
    - Retrieve all tasks with date info
    - Render in calendar format
8. If user sets **Reminder**:
    - Schedule notification based on due date/time
9. If it's time for a **Notification**:
    - Trigger reminder popup or system alert
10. On **Exit**:
    - Save any unsaved changes
    - Close database connection
    - Exit the application
11. **End**

Pseudocode:

1. Main Program Initialization:

```
    BEGIN
  Initialize GUI using JavaFX
  Connect to MySQL Database using JDBC
      LOAD all tasks from database
      DISPLAY Home Screen with:
        - Task List
        - Calendar View
        - Navigation Options (Add Task, Edit, Delete, View Calendar)
END
```

2. Add New Task:

```
    FUNCTION addTask()
      PROMPT user for:
        - Task Title
        - Description
        - Due Date
        - Priority (High/Medium/Low)
        - Category (Optional)
        - Reminder Time (Optional)

      IF input is valid THEN
        STORE task in MySQL database
        UPDATE task list in GUI
        IF reminder is set THEN
          SCHEDULE reminder in background
      ELSE
        DISPLAY error message
      ENDIF
    END FUNCTION
```

3. Edit Existing Task:

```
    FUNCTION editTask(taskID)
      RETRIEVE task FROM database using taskID
      DISPLAY existing task details to user
      PROMPT user for updated information

      IF input is valid THEN
        UPDATE task in database
        REFRESH task list in GUI
      ELSE
        DISPLAY error message
      ENDIF
    END FUNCTION
```

4. Delete Task:

```
FUNCTION deleteTask(taskID)
  ASK for confirmation
  IF confirmed THEN
    DELETE task FROM database
    REMOVE task FROM GUI list
  ENDIF
END FUNCTION
```

5. View Calendar:

```
FUNCTION viewCalendar()
  FETCH all tasks with due dates
  GROUP tasks by date
  DISPLAY calendar view with tasks shown on respective dates
END FUNCTION
```

6. Reminder Scheduler:

```
FUNCTION startReminderService()
  WHILE application is running DO
    GET current system time
    FOR each task in database DO
      IF current time == task.reminderTime THEN
        DISPLAY notification popup
      ENDIF
    ENDFOR
    WAIT for 1 minute
  ENDWHILE
END FUNCTION
```

7. Exit Application:

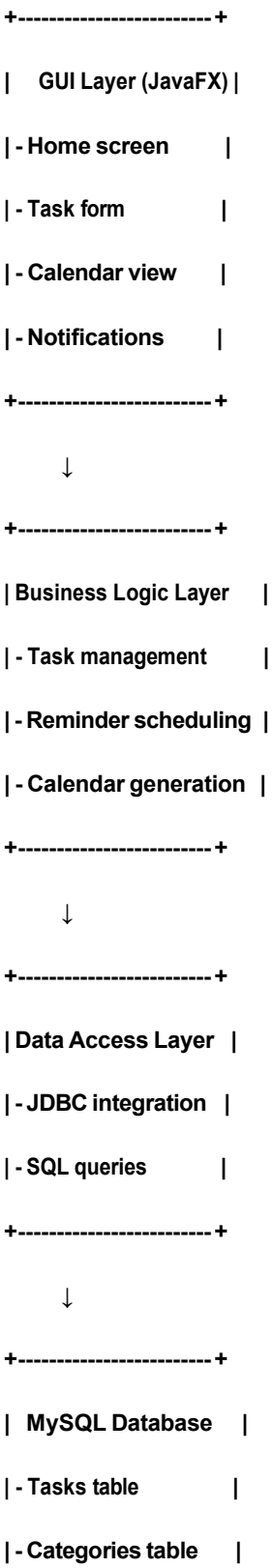
```
FUNCTION exitApplication()
  CLOSE database connection
  SAVE any temporary data if needed
  TERMINATE application
END FUNCTION
```

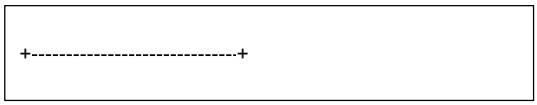
Modular Breakdown:

Functionality	Module	Tech Used
GUI	JavaFX	JavaFX Components (Scene, Stage, etc.)
Data Storage	MySQL + JDBC	SQL, JDBC
Task Scheduling	TaskManager class	Java Collections
Reminders	Background Thread/Timer	Java Timer/Thread
Calendar	CalendarView class	JavaFX GridPane or Custom View

5. System Design

System Architecture:





The architecture follows a **three-layer design**:

1. **Presentation Layer (Frontend)** – Java Swing or console interface for input/output.
2. **Logic Layer (Business Logic)** – Java classes handling skill matching and reallocation.
3. **Database Layer (Backend)** – MySQL database to store and update records.

6. System Requirements

Software Requirements:

- **Operating System:** Windows 10 / 11 or Linux
- **Programming Language:** Java (JDK 8 or higher)
- **Database:** MySQL
- **IDE:** Eclipse / NetBeans / IntelliJ IDEA
- **Connector:** MySQL JDBC Driver

Hardware Requirements:

- **Processor:** Intel i5 or higher
- **RAM:** Minimum 8 GB
- **Storage:** Minimum 500 GB
- **Display:** 1366 × 768 resolution

7. Implementation Details

The **Digital Assist Organizer** is a Java desktop application that integrates a powerful backend (MySQL + JDBC), a sleek GUI (JavaFX), and robust business logic to provide an all-in-one productivity tool. By modularizing the application into clear layers and components, it remains maintainable, scalable, and extensible for diverse user needs — from students and developers to working professionals.

Key modules include:

1. **Language Used:** Java
2. **GUI Framework:** JavaFX for user interface
3. **Database:** MySQL for persistent task storage
4. **Database Connectivity:** JDBC for communication between Java and MySQL
5. **Task Management:** Create, update, delete, and prioritize tasks
6. **Reminder System:** Scheduled reminders using Java Timer or ScheduledExecutorService
7. **Calendar View:** Built using JavaFX components (GridPane, DatePicker) to show tasks by date
8. **Data Persistence:** Tasks stored in a tasks table with fields like title, description, due date, priority, and status
9. **Modular Architecture:** Separation of GUI, business logic, and data access layers
10. **Responsive UI:** Clean and user-friendly design suitable for students, developers, and professionals
11. **Notifications:** Alerts for upcoming or overdue tasks
12. **Filtering s Search:** View tasks by date, category, or priority

PROGAM CODE:

```
import com.mongodb.MongoTimeoutException;
import com.mongodb.client.*;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Updates;
import org.bson.Document;
import org.bson.types.ObjectId;

import java.time.LocalDate;
import java.util.Scanner;

public class DigitalAssistOrganizer {
  private static final String MONGODB_URI = "mongodb://localhost:27017"; // Change if
needed
  private static final String DB_NAME = "digital_assist";

  private MongoClient mongoClient;
  private MongoDatabase db;
  private MongoCollection<Document> tasksCol;
  private MongoCollection<Document> eventsCol;
  private MongoCollection<Document> notesCol;
  private boolean dbConnected = false;

  // Constructor with MongoDB connection safety
  public DigitalAssistOrganizer() {
    try {
```

```
            mongoClient = MongoClients.create(MONGODB_URI);
            db = mongoClient.getDatabase(DB_NAME);
            tasksCol = db.getCollection("tasks");
            eventsCol = db.getCollection("events");
            notesCol = db.getCollection("notes");
            dbConnected = true;
            System.out.println("✅ Connected to MongoDB successfully.");
        } catch (MongoTimeoutException e) {
            System.out.println("⚠️ Unable to connect to MongoDB. Please make sure MongoDB server is running on localhost:27017.");
        } catch (Exception e) {
            System.out.println("⚠️ Unexpected error while connecting to MongoDB: " + e.getMessage());
        }
    }

    public void close() {
        if (mongoClient != null) mongoClient.close();
    }

    private boolean checkConnection() {
        if (!dbConnected) {
            System.out.println("❌ MongoDB is not connected. Please start MongoDB and restart the application.");
            return false;
        }
```

```
    return true;
}

// ---------- TASKS ----------
public String addTask(String title, String description, LocalDate dueDate) {
    if (!checkConnection()) return null;
    try {
        Document d = new Document("title", title)
                .append("description", description)
                .append("dueDate", dueDate != null ? dueDate.toString() : null)
                .append("completed", false)
                .append("createdAt", java.time.Instant.now().toString());
        tasksCol.insertOne(d);
        return d.getObjectId("_id").toHexString();
    } catch (Exception e) {
        System.out.println("❌ Failed to add task: " + e.getMessage());
        return null;
    }
}

public void listTasks() {
    if (!checkConnection()) return;
    System.out.println("---- Tasks ---- ");
    for (Document d : tasksCol.find()) {
        printDoc(d);
    }
```

```
}

public boolean completeTask(String idHex) {
  if (!checkConnection()) return false;
  try {
    ObjectId id = new ObjectId(idHex);
    var res = tasksCol.updateOne(Filters.eq("_id", id),Updates.set("completed",true));
    return res.getModifiedCount() > 0;
  } catch (Exception e) {
    System.out.println("❌ Failed to complete task: " + e.getMessage());
    return false;
  }
}

public boolean deleteTask(String idHex) {
  if (!checkConnection()) return false;
  try {
    ObjectId id = new ObjectId(idHex);
    var res = tasksCol.deleteOne(Filters.eq("_id", id));
    return res.getDeletedCount() > 0;
  } catch (Exception e) {
    System.out.println("❌ Failed to delete task: " + e.getMessage());
    return false;
  }
}
```

```
// ---------- EVENTS ----------
public String addEvent(String title, String description, LocalDate date) {
  if (!checkConnection()) return null;
  try {
     Document d = new Document("title", title)
          .append("description", description)
          .append("date", date != null ? date.toString() : null)
          .append("createdAt", java.time.Instant.now().toString());
     eventsCol.insertOne(d);
     return d.getObjectId("_id").toHexString();
  } catch (Exception e) {
     System.out.println("❌ Failed to add event: " + e.getMessage());
     return null;
  }
}

public void listEvents() {
  if (!checkConnection()) return;
  System.out.println("---- Events --");
  for (Document d : eventsCol.find()) {
     printDoc(d);
  }
}

public boolean deleteEvent(String idHex) {
  if (!checkConnection()) return false;
```

```
    try {
      ObjectId id = new ObjectId(idHex);
      var res = eventsCol.deleteOne(Filters.eq("_id", id));
      return res.getDeletedCount() > 0;
    } catch (Exception e) {
      System.out.println("❌ Failed to delete event: " + e.getMessage());
      return false;
    }
  }

  // ---------- NOTES ----------
  public String addNote(String title, String content) {
    if (!checkConnection()) return null;
    try {
      Document d = new Document("title", title)
          .append("content", content)
          .append("createdAt", java.time.Instant.now().toString());
      notesCol.insertOne(d);
      return d.getObjectId("_id").toHexString();
    } catch (Exception e) {
      System.out.println("❌ Failed to add note: " + e.getMessage());
      return null;
    }
  }

  public void listNotes() {
```

```
    if (!checkConnection()) return;
    System.out.println("---- Notes -- ");
    for (Document d : notesCol.find()) {
        printDoc(d);
    }
}

public boolean deleteNote(String idHex) {
    if (!checkConnection()) return false;
    try {
        ObjectId id = new ObjectId(idHex);
        var res = notesCol.deleteOne(Filters.eq("_id", id));
        return res.getDeletedCount() > 0;
    } catch (Exception e) {
        System.out.println("❌ Failed to delete note: " + e.getMessage());
        return false;
    }
}

// ---------------- Utility ----------------
private void printDoc(Document d) {
    String id = d.getObjectId("_id").toHexString();
    System.out.println("ID: " + id);
    for (String key : d.keySet()) {
        if ("_id".equals(key)) continue;
        System.out.println("  " + key + ": " + d.get(key));
```

```
    }
    System.out.println();
  }

  // ---------------- CLI ----------------
  public static void main(String[] args) {
    try (Scanner scanner = new Scanner(System.in)) {
      DigitalAssistOrganizer dao = new DigitalAssistOrganizer();
      System.out.println("Welcome to Digital Assist Organizer!");

      boolean running = true;
      while (running) {
        printMenu();
        System.out.print("Choose option: ");
        String opt = scanner.nextLine().trim();
        switch (opt) {
          case "1":
            System.out.print("Title: ");
            String t = scanner.nextLine();
            System.out.print("Description: ");
            String td = scanner.nextLine();
            System.out.print("Due date (YYYY-MM-DD) or blank: ");
            String due = scanner.nextLine().trim();
            LocalDate dueDate = due.isBlank() ? null : LocalDate.parse(due);
            String taskId = dao.addTask(t, td, dueDate);
            if (taskId != null)
```

```
                System.out.println("✅ Task created with id: " + taskId);
                break;
            case "2":
                dao.listTasks();
                break;
            case "3":
                System.out.print("Task ID to mark complete: ");
                String cid = scanner.nextLine().trim();
                System.out.println(dao.completeTask(cid) ? "✅ Marked complete." : "❌
Failed — check ID.");
                break;
            case "4":
                System.out.print("Task ID to delete: ");
                String del = scanner.nextLine().trim();
                System.out.println(dao.deleteTask(del) ? "✅ Deleted." : "❌ Failed — check
ID.");
                break;
            case "5":
                System.out.print("Event title: ");
                String et = scanner.nextLine();
                System.out.print("Description: ");
                String ed = scanner.nextLine();
                System.out.print("Date (YYYY-MM-DD) or blank: ");
                String date = scanner.nextLine().trim();
                LocalDate evDate = date.isBlank() ? null : LocalDate.parse(date);
                String evId = dao.addEvent(et, ed, evDate);
```

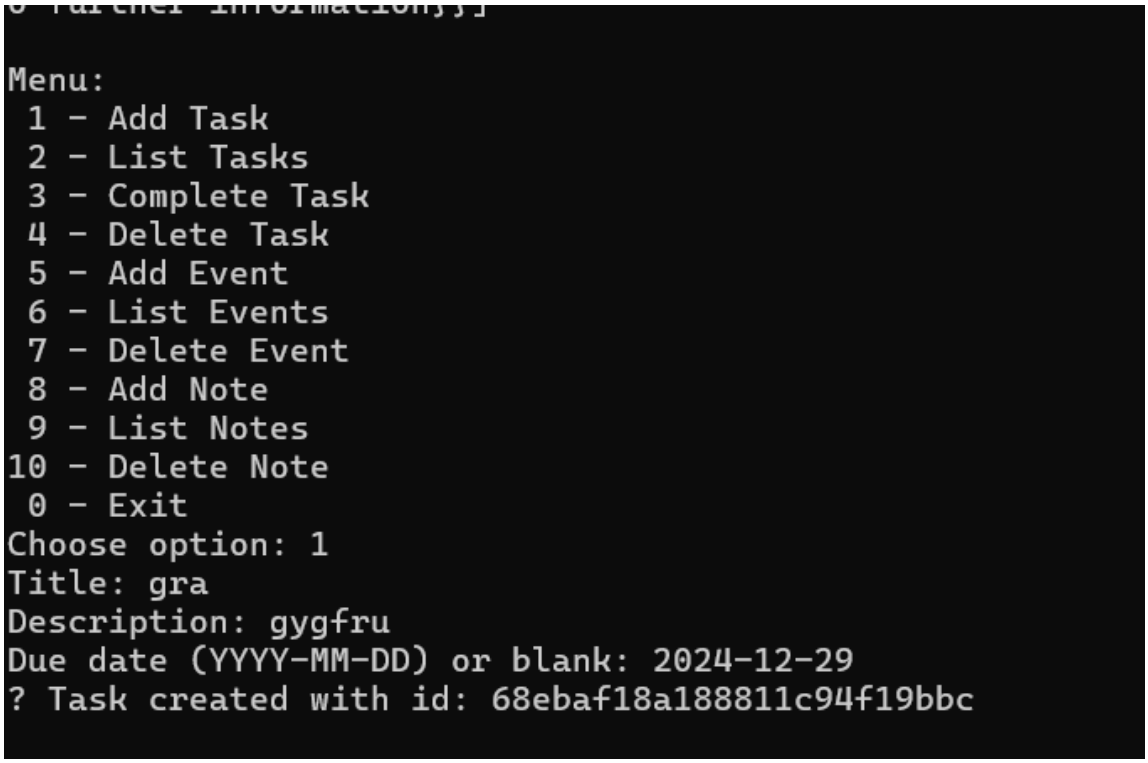
```
            if (evId != null)
              System.out.println("✅ Event created with id: " + evId);
            break;
          case "6":
            dao.listEvents();
            break;
          case "7":
            System.out.print("Event ID to delete: ");
            String edel = scanner.nextLine().trim();
            System.out.println(dao.deleteEvent(edel) ? "✅ Deleted." : "❌ Failed — check ID.");
            break;
          case "8":
            System.out.print("Note title: ");
            String nt = scanner.nextLine();
            System.out.print("Content: ");
            String nc = scanner.nextLine();
            String nId = dao.addNote(nt, nc);
            if (nId != null)
              System.out.println("✅ Note created with id: " + nId);
            break;
          case "G":
            dao.listNotes();
            break;
          case "10":
            System.out.print("Note ID to delete: ");
```

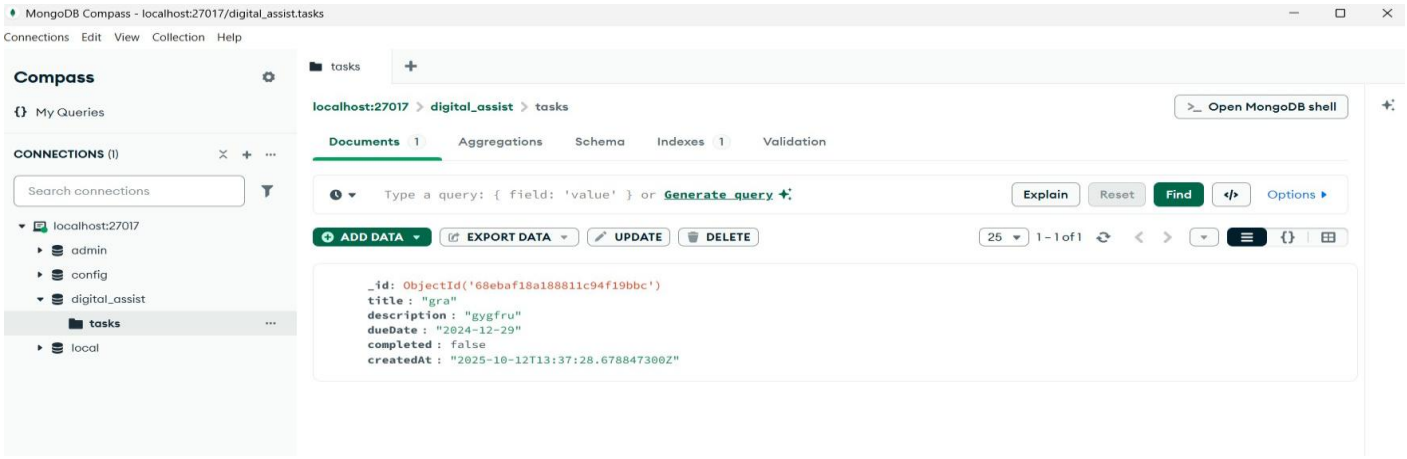
```
                String ndel = scanner.nextLine().trim();
                System.out.println(dao.deleteNote(ndel) ? "✅ Deleted." : "❌ Failed — check ID.");
                break;
            case "0":
                running = false;
                break;
            default:
                System.out.println("⚠️ Invalid option");
        }
      }
      dao.close();
      System.out.println("👋 Goodbye!");
    } catch (Exception ex) {
      System.out.println("❌ Unexpected error: " + ex.getMessage());
    }
  }

  private static void printMenu() {
    System.out.println("\nMenu:");
    System.out.println(" 1 - Add Task");
    System.out.println(" 2 - List Tasks");
    System.out.println(" 3 - Complete Task");
    System.out.println(" 4 - Delete Task");
    System.out.println(" 5 - Add Event");
    System.out.println(" 6 - List Events");
```

```
        System.out.println(" 7 - Delete Event");
        System.out.println(" 8 - Add Note");
        System.out.println(" G - List Notes");
        System.out.println("10 - Delete Note");
        System.out.println(" 0 - Exit");
    }
}
```

8. Output and Results

Expected Output Screens:





The results demonstrate that system efficiently identifie suitable employee and provides HR manager with actionable reallocation suggestions.

9. Conclusion

The **Digital Assist Organizer** successfully addresses the growing need for effective task and time management among students, developers, and professionals. By leveraging Java for core development, JavaFX for responsive user interface, MySQL for reliable data storage, the application provides a simple yet powerful solution to organize daily responsibilities.

Through features like task creation, calendar scheduling, priority setting, and automated reminders, users can efficiently plan their activities, reduce missed deadlines, boost overall productivity. The modular and scalable design ensures the application can be extended with future enhancements such as user authentication, cloud sync, or mobile support.

Overall, this project demonstrates the practical integration of object-oriented programming, user-centric design, and database management to solve real-world organizational challenges in a digital environment.

10. References

1. Oracle Java Documentation
https://docs.oracle.com/javase/
(Official documentation for Java SE APIs and libraries)

2. JavaFX Documentation – Oracle
https://openjfx.io/
(Reference for JavaFX components, layouts, and UI development)

3. MySQL Documentation – Oracle
https://dev.mysql.com/doc/
(For database design, SQL queries, and JDBC integration)

4. GeeksforGeeks – Java Tutorials
https://www.geeksforgeeks.org/java/
(Concepts on Java programming, JDBC, OOP principles)

5. W3Schools – SQL Tutorial
https://www.w3schools.com/sql/
(Basic to advanced SQL queries used for backend data management)

6. GitHub – Open-source Java Projects
https://github.com/
(For code examples, project inspiration, and version control usage)