ACADSYNC

SMART ACADEMIC TIMETABLE GENERATOR

MEMBERS

1 AKSHARA P

2 ANUSREE M

3 ASHIL SURESH

4 SREYAS P

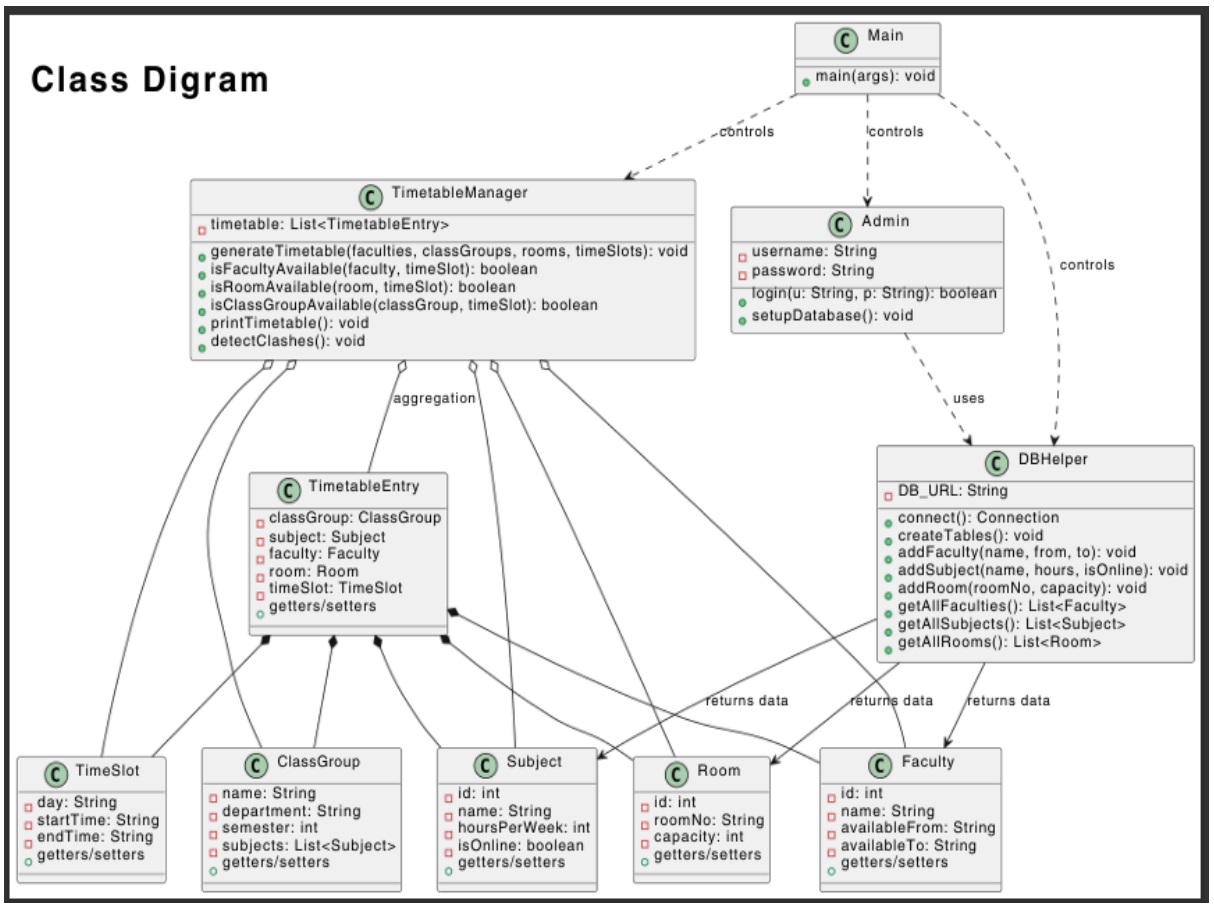
ABSTRACT

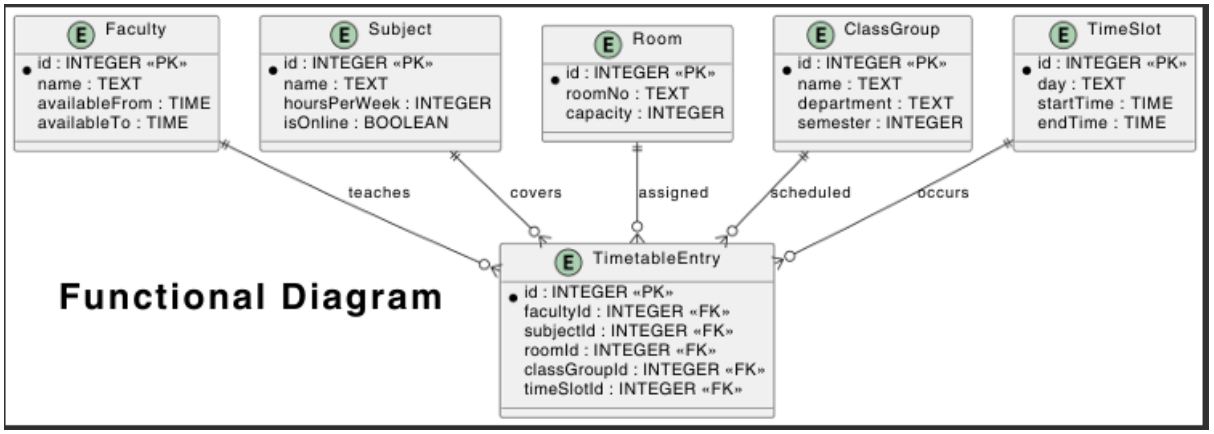
AcadSync is a Java-based intelligent academic timetable generator designed to automate and optimize the scheduling process in educational institutions. Built using Object-Oriented Programming (OOP) principles, it efficiently manages various constraints such as faculty availability, room allocation, subject workload, and user preferences to produce conflict-free timetables.

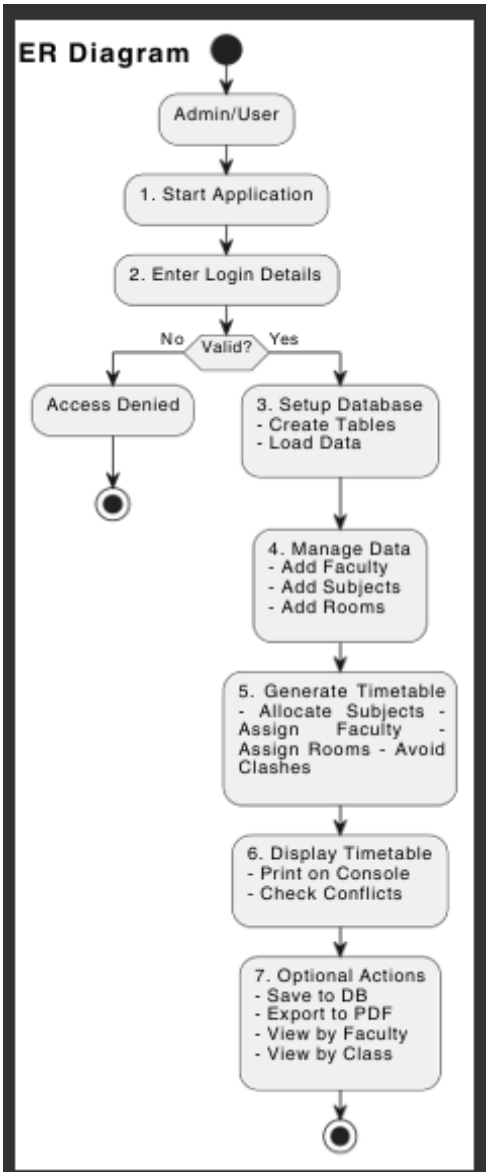
The system enables users to generate, modify, and view schedules by class, faculty, or room, offering flexibility and accuracy in planning. AcadSync incorporates advanced features like automatic conflict detection, dynamic timetable regeneration, workload balancing, holiday management, and support for online/offline sessions.

An intuitive GUI built with Java Swing ensures user-friendly interaction, while SQLite database integration via JDBC provides reliable and persistent data storage. Additional utilities such as PDF/Excel export, analytics dashboard, and secure multi-user login enhance the system's practicality.

By combining modular design with efficient algorithmic logic, AcadSync demonstrates the effective application of inheritance, encapsulation, abstraction, and polymorphism in solving real-world scheduling challenges, making it a scalable and robust solution for schools, colleges, and coaching institutions.


Class Digram
Main
main(args): void
controls
controls
controls
TimetableManager
timetable: List<TimetableEntry>
generateTimetable(faculties, classGroups, rooms, timeSlots): void
isFacultyAvailable(faculty, timeSlot): boolean
isRoomAvailable(room, timeSlot): boolean
isClassGroupAvailable(classGroup, timeSlot): boolean
printTimetable(): void
detectClashes(): void
Admin
username: String
password: String
login(u: String, p: String): boolean
setupDatabase(): void
aggregation
uses
TimetableEntry
classGroup: ClassGroup
subject: Subject
faculty: Faculty
room: Room
timeSlot: TimeSlot
getters/setters
DBHelper
DB_URL: String
connect(): Connection
createTables(): void
addFaculty(name, from, to): void
addSubject(name, hours, isOnline): void
addRoom(roomNo, capacity): void
getAllFaculties(): List<Faculty>
getAllSubjects(): List<Subject>
getAllRooms(): List<Room>
returns data
returns data
returns data
TimeSlot
day: String
startTime: String
endTime: String
getters/setters
ClassGroup
name: String
department: String
semester: int
subjects: List<Subject>
getters/setters
Subject
id: int
name: String
hoursPerWeek: int
isOnline: boolean
getters/setters
Room
id: int
roomNo: String
capacity: int
getters/setters
Faculty
id: int
name: String
availableFrom: String
availableTo: String
getters/setters


Faculty
id : INTEGER «PK»
name : TEXT
availableFrom : TIME
availableTo : TIME
Subject
id : INTEGER «PK»
name : TEXT
hoursPerWeek : INTEGER
isOnline : BOOLEAN
Room
id : INTEGER «PK»
roomNo : TEXT
capacity : INTEGER
ClassGroup
id : INTEGER «PK»
name : TEXT
department : TEXT
semester : INTEGER
TimeSlot
id : INTEGER «PK»
day : TEXT
startTime : TIME
endTime : TIME
teaches
covers
assigned
scheduled
occurs
Functional Diagram
TimetableEntry
id : INTEGER «PK»
facultyId : INTEGER «FK»
subjectId : INTEGER «FK»
roomId : INTEGER «FK»
classGroupId : INTEGER «FK»
timeSlotId : INTEGER «FK»


ER Diagram
Admin/User
1. Start Application
2. Enter Login Details
No
Valid?
Yes
Access Denied
3. Setup Database
- Create Tables
- Load Data
4. Manage Data
- Add Faculty
- Add Subjects
- Add Rooms
5. Generate Timetable
- Allocate Subjects - Assign Faculty - Assign Rooms - Avoid Clashes
6. Display Timetable
- Print on Console
- Check Conflicts
7. Optional Actions
- Save to DB
- Export to PDF
- View by Faculty
- View by Class

SYSTEM MODULES:

a. User Management – Login, role-based access (Admin/Faculty)
b. Faculty & Subject Management – Add/edit faculty & subjects, assign subjects
c. Class & Room Management – Manage classes, rooms, capacities, preferences
d. TimeSlot & Schedule – Define slots, allocate classes, detect clashes
e. Timetable Generation – Core scheduling logic, dynamic regeneration
f. Preferences & Rules Engine – Apply rules, optimize schedules
g. Analytics & Reporting – Faculty load, room usage, export PDF/Excel
h. Notification Module – Alerts for changes or conflicts
i. Database & Persistence – CRUD operations via SQLite
j. User Interface (GUI/Console) – Dashboards and timetable display

SYSTEM ARCHITECTURE

a. **Presentation Layer** → Java Swing GUI

b. Business Logic Layer → Core Java Classes

c. **Data Access Layer** → JDBC Operations

d. **Database Layer** → SQLite

Workflow Example:
When a user generates a timetable, the system checks faculty and room availability through the core logic, detects conflicts, saves the final schedule to the SQLite database, and displays it instantly on the GUI.

PROTOTYPE CREATION

We have successfully built a prototype of AcadSync: A Smart Academic Timetable Generator. The prototype demonstrates the core functionalities of the system, including:

- **Login System:** Secure role-based access for Admin and Faculty.
- **Data Management:** Adding and managing faculty, subjects, classes, and rooms.
- **Basic Timetable Generation:** Generates a conflict-free timetable for sample input data.
- **Data Persistence:** Stores and retrieves information using SQLite.
- **User Interaction:** Simple console or GUI interface to input data and view timetables.

This prototype was shared with the team to gather feedback before the coding session.

SOURCE CODE DEVELOPMENT

```
AcadSync/
├── lib/
│   └── sqlite-jdbc-3.50.3.0.jar
│
├── src/
│   ├── core/
│   │   └── TimetableManager.java
│   │
│   ├── gui/
│   │   ├── AllRoomsScheduleView.java
│   │   ├── DashboardGUI.java
│   │   ├── DayWiseScheduleView.java
│   │   ├── FacultyEditForm.java
│   │   ├── FacultyForm.java
│   │   ├── FacultyListView.java
│   │   ├── FacultyManagement.java
│   │   ├── LoginGUI.java
│   │   ├── RoomForm.java
│   │   ├── RoomManagement.java
│   │   ├── RoomTimetableView.java
│   │   ├── SubjectForm.java
│   │   ├── SubjectManagement.java
│   │   ├── TimetableView.java
│   │
│   ├── models/
│   │   ├── Admin.java
│   │   ├── ClassGroup.java
│   │   ├── Faculty.java
│   │   ├── Room.java
│   │   ├── Subject.java
│   │   ├── TimeSlot.java
│   │   └── TimetableEntry.java
│   │
│   ├── utils/
│   │   └── DBHelper.java
│   │
│   └── Main.java
```

```
│
├── acadsync.db
├── README.md
├── LICENSE
└── .gitignore
```

TECHNOLOGIES USED:

Programming Language: Java (Core + Swing/AWT) — for logic, GUI, and database connectivity.

Database: SQLite — lightweight, stores data in a single `.db` file, accessed via JDBC.

JDBC: `sqlite-jdbc-3.50.3.0.jar` used to connect Java with SQLite for queries and data management.

OOP Concepts: Encapsulation, Inheritance, Polymorphism, Abstraction — organized across models, core logic, and GUI.

GUI Framework: Java Swing/AWT — for interactive forms and dashboards (JFrame, JPanel, JTable, etc.).

Development Environment: VS Code with Java extensions.

Version Control: Git — manages source code and excludes build/database files via `.gitignore`.

Documentation: README.md and LICENSE for setup instructions and usage right

AcadSync - Internal Test Case Report for Prototype Submission

Project Name: AcadSync - Timetable Management System **Submission Date:** 09/09/2025
Prepared By:

Module 1: Admin Login

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result
TC1	Test empty fields	Username: "", Password: ""	Show error: "Please enter both username and password"	Show error: "Please enter both username and password"	Pass
TC2	Test invalid credentials	Username: "john", Password: "wrongpass"	Show error: "Invalid credentials"	Show error: "Invalid credentials"	Pass
TC3	Test valid login	Username: "john", Password: "correctpass"	Login successful → Dashboard opens	Login successful → Dashboard opens	Pass
TC4	Test new admin creation	Username: "johni", Password: "1234"	"Admin account created successfully"	"Admin account created successfully"	Pass

Module 2: Add Subject

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result
TC5	Add a new subject successfully	Subject Name="OOP", Hours/Week="5"	"Subject added successfully"	"Subject added successfully"	Pass
TC6	Add a duplicate subject name	Subject Name="OOP", Hours/Week="3"	"Subject already exists"	"Subject already exists"	Pass
TC7	Test with empty fields	Subject Name="", Hours/Week=""	Show error: "Please fill in all fields"	Show error: "Please fill in all fields"	Pass

Module 3: Add Faculty

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result
TC8	Add a new faculty member successfully	Faculty Name: "John Doe", Available From: "9", Available To: "17"	"Faculty added successfully"	"Faculty added successfully"	Pass
TC9	Assign a single subject	Faculty Name: "John Doe", select "Maths", click Add	"Maths" moves to Assigned Subjects list	"Maths" moves to Assigned Subjects list	Pass

TC10	Assign all subjects	Faculty Name: "John Doe", click Add All	All subjects move to Assigned Subjects list	All subjects move to Assigned Subjects list	Pass
TC11	Remove a single subject	Faculty Name: "John Doe", select "Maths", click Remove	"Maths" moves back to Available Subjects list	"Maths" moves back to Available Subjects list	Pass
TC12	Remove all subjects	Faculty Name: "John Doe", click Remove All	All subjects move back to Available Subjects list	All subjects move back to Available Subjects list	Pass
TC13	Add faculty with duplicate name	Faculty Name: "John Doe", Available From: "9", Available To: "17"	"Faculty already exists"	"Faculty already exists"	Pass

Module 4: Add Room

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result

TC16	Add a new room successfully	Room No: "101", Capacity: "50"	"Room added successfully"	"Room added successfully"	Pass
TC17	Add a room with a duplicate number	Room No: "101", Capacity: "45"	"Room already exists"	"Room already exists"	Pass
TC18	Add a room with non-numeric capacity	Room No: "102", Capacity: "abc"	Show error: "Capacity must be a number"	Show error: "Capacity must be a number"	Pass

Module 5: Manage Faculty (Edit & Delete)

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result
TC19	Edit faculty name successfully	Select "John Doe", change name to "John A. Doe"	"Faculty updated successfully"	"Faculty updated successfully"	Pass
TC20	Delete a faculty member	Select "Jane Doe", click Delete	"Faculty deleted successfully"	"Faculty deleted successfully"	Pass
TC21	Attempt to delete faculty with subjects	Select "John A. Doe", click Delete	"Cannot delete faculty, they have assigned subjects"	"Cannot delete faculty, they have assigned subjects"	Pass

Module 6: Manage Subject (Edit & Delete)

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result
TC22	Edit subject hours successfully	Select "OOP", change hours to "4"	"Subject updated successfully"	"Subject updated successfully"	Pass
TC23	Delete a subject	Select "Data Structures", click Delete	"Subject deleted successfully"	"Subject deleted successfully"	Pass
TC24	Attempt to delete subject assigned	Select "OOP", click Delete	"Cannot delete subject, it is assigned to a faculty"	"Cannot delete subject, it is assigned to a faculty"	Pass

Module 7: Manage Room (Edit & Delete)

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result
TC25	Edit room capacity successfully	Select "101", change capacity to "55"	"Room updated successfully"	"Room updated successfully"	Pass
TC26	Delete a room	Select "102", click Delete	"Room deleted successfully"	"Room deleted successfully"	Pass
TC27	Attempt to delete a room in use	Select "101", click Delete	"Cannot delete room, it is in use"	"Cannot delete room, it is in use"	Pass

Module 8: View Modules (Faculty & Assignments, Timetable, Room Timetable)

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result
TC28	View all faculty assignments	Click "View Faculty and Assignment"	A table showing all faculty and their assigned subjects is displayed	A table showing all faculty and their assigned subjects is displayed	Pass
TC29	Generate a timetable with valid data	Click "Generate Timetable"	A complete timetable is generated and displayed	A complete timetable is generated and displayed	Pass
TC30	Attempt to generate with missing data	Click "Generate Timetable" (no rooms)	"Error: Required data is missing"	"Error: Required data is missing"	Pass
TC31	View generated timetable	Click "View Timetable"	The last generated timetable is displayed	The last generated timetable is displayed	Pass
TC32	View specific room timetable	Select Room "101", click "View Room Timetable"	A timetable specifically for room "101" is displayed	A timetable specifically for room "101" is displayed	Pass

Module 9: Logout

Test Case ID	Test Description	Input	Expected Output	Actual Output	Result

TC33	Successful logout	Click "Logout" button	User is logged out and redirected to the Login page	User is logged out and redirected to the Login page	Pass
TC34	Access dashboard after logout	Navigate back to the Dashboard URL	User is redirected to the Login page	User is redirected to the Login page	Pass