# EVENT SCHEDULING SYSTEM

**A PROJECT REPORT**

*Submitted by*

**Aarsh Sharma – 23BCS11331**
**Paarth Khera -23BCS11422**
**K Dhiraj – 23BCS11477**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

**IN**

COMPUTER SCIENCE & ENGINEERING

**Chandigarh University**

Month, 2025

# TABLE OF CONTENT

# CHAPTER 1: INTRODUCTION

## *1.1. Introduction to Project*

The Event Registration System is an integrated platform developed using both Command Line Interface (CLI) and Graphical User Interface (GUI) designs. It is designed to manage the registration process for events in academic institutions. The application allows students to browse and register for events, while administrators can manage event details and view participant lists.

The CLI version provides a lightweight, terminal-based interface suitable for environments where graphical support is limited. In contrast, the GUI version (developed using JavaFX or Swing) enhances user experience by offering intuitive interactions, clickable buttons, and form-based navigation.

This dual-interface approach makes the system flexible, catering to a broader range of use cases. It serves as a robust tool for internal event management and provides a practical learning experience in handling different types of user interfaces.

## *1.2. Identification of Problem*

Traditional paper-based and spreadsheet-based registration systems lack efficiency, scalability, and reliability. Users face difficulties in accessing updated data, and administrators often find it hard to manage increasing volumes of participant information.

While CLI applications provide fast and resource-efficient solutions, they may not be user-friendly for everyone. Hence, there is a need for a dual-interface solution that offers both CLI efficiency and GUI usability. This project aims to fill that gap by designing a hybrid system that can cater to technical users and non-technical participants alike.

# CHAPTER 2: BACKGROUND STUDY

## *2.1. Existing Solutions*

Various platforms exist for event registration, including ERP modules, online survey tools, and commercial event management software. However, most require continuous internet connectivity, cloud integration, and paid licenses.

Web and GUI-based systems offer accessibility but may be resource-intensive. On the other hand, CLI applications run on minimal resources but often lack an intuitive design. This project explores a hybrid approach where CLI and GUI versions co-exist, providing flexibility to users depending on their environment and comfort level.

## *2.2. Problem Definition*

This project aims to implement a robust event registration system with both CLI and GUI support. The application should allow students and administrators to interact with the system through either interface seamlessly. Key features should include:

- Registration and deregistration
- Event creation, updating, and deletion
- Data persistence with relational databases

Both interfaces must maintain consistent backend logic and use shared database models to ensure uniformity and data integrity.

## *2.3. Goals/Objectives*

- Develop CLI and GUI versions of the event registration system.
- Use object-oriented design to share backend logic across both interfaces.
- Ensure cross-platform compatibility and user-friendly design.
- Store and retrieve data from a persistent SQL database.
- Provide error handling, input validation, and authentication mechanisms.

# CHAPTER 3: DESIGN FLOW/PROCESS

## *3.1. Evaluation & Selection of Specifications/Features*

Java was selected for its robustness, OOP capabilities, and cross-platform nature. For CLI interaction, standard input/output via the console was utilized, while the GUI was built using JavaFX, enabling structured layouts and visual widgets.

SQLite was chosen for persistent data storage due to its simplicity and integration ease. Features like student registration, event creation, and admin authentication were included in both CLI and GUI.

## *3.2. Analysis of Features and Finalization Subject to Constraints*

The dual-interface system had to be designed with shared backend services to prevent redundancy. Key modules like `DB.java` and business logic classes were shared by both CLI and GUI layers. Constraints such as development time, interface complexity, and user testing influenced the final feature set.

The CLI was prioritized for rapid testing and core functionality, while the GUI added a layer of usability. Both interfaces provided complete access to event management, ensuring flexibility without compromising on system features.

## 3.1. *Design Flow*

The development of the Event Registration System followed a systematic, modular, and incremental approach. Each component was built and tested in a sequence to ensure reliability, reusability, and maintainability.

We began by creating DB.java, which formed the foundation for establishing the database connectivity between Java and SQLite using the JDBC driver. This file was responsible for managing all database-related tasks such as executing queries, handling connections, and performing data manipulations like inserts, updates, deletions, and retrievals. It was important to complete this file early in development so that all future components could rely on an active database connection and consistent data handling logic.

Once the connectivity was functional and database operations could be tested through Java methods, we proceeded to build the entity classes: Event.java, Student.java, and Registration.java. These classes were designed as Plain Old Java Objects (POJOs) to represent and encapsulate the data for events, students, and registrations respectively. Each class was designed with relevant attributes and accessor methods, ensuring that they could interact cleanly with the database layer.

Next, Main.java was developed as the core CLI driver class. It handled user interaction through the command line, allowing administrators and students to interact with the system using menu-driven input. It invoked the relevant functions from the entity classes and DB handler to carry out the desired operations. This formed the CLI interface and allowed us to thoroughly test system logic and features.

With the CLI logic validated, we turned to building the graphical interface. JavaFX was used to create GUI components such as login screens, registration forms, and dashboards. FXML files defined the layout, while controller classes contained event-handling logic. These controllers communicated with the same backend (DB.java and POJOs), ensuring that both CLI and GUI shared consistent functionality.

Lastly, schema.sql was written to define the relational database structure. This included table definitions for events, students, and registrations, along with the necessary constraints such as primary and foreign keys. Since the DB.java class was already functional, the schema.sql file was used to initialize the actual database structure during deployment or reinstallation, ensuring the design matched the operational code.

This progression—from DB connectivity, to business logic classes, to CLI interface, then GUI, and finally schema setup—ensured a smooth development flow and minimized integration issues.

# CHAPTER 4: RESULTS ANALYSIS AND VALIDATION

## *4.1. Implementation of Solution*

The project implementation was done in two phases. The CLI version was completed first, with key modules like Event.java, Student.java, Registration.java, and DB.java. The CLI interface was managed through Main.java, allowing text-based navigation.

Later, the GUI was developed using JavaFX. It introduced scenes for login, event management, and student registration. Scene controllers invoked the same backend classes used in the CLI version.

Testing was performed for both interfaces. Edge cases like invalid input, duplicate registration, and empty event lists were handled gracefully. The system was validated for responsiveness, accuracy, and data integrity across CLI and GUI modes.

---

# CHAPTER 5: CONCLUSION AND FUTURE WORK

## *5.1. Conclusion*

The Event Registration System with both CLI and GUI support successfully meets the needs of academic event management. The CLI offers speed and simplicity, ideal for developers and technical administrators. The GUI adds a user-friendly interface suitable for everyday users.

The dual-interface model demonstrates the power of modular backend design. It enables the coexistence of multiple interfaces over a shared codebase. This approach makes the system scalable, adaptable, and more accessible to a wide range of users.

Moreover, the system not only provides hands-on experience with technologies such as Java, SQLite, JDBC, and JavaFX but also helps develop practical knowledge in designing full-stack applications. Students using this system gain real-world exposure to object-oriented design, modular coding practices, UI/UX principles, and database handling.

The implementation of both command-line and graphical interfaces allows us to evaluate user behavior and preference under different interaction modalities. It broadens the project's scope, as it caters to both resource-constrained environments and general users seeking ease of use.In conclusion, this project exemplifies the successful integration of backend logic with diverse front-end access points. It stands as a viable model for building efficient and adaptable systems that balance performance and usability effectively.

## *5.2. Future Work*

Future developments can include:

- Integrating web-based frontend using HTML/CSS/JavaScript with REST APIs.
- Adding advanced filters and search in the GUI.
- Creating mobile app versions for on-the-go access.
- Incorporating event analytics and real-time dashboards.
- Enhancing security with hashed credentials and OTP logins.