Faculty	**Engineering and Technology**		
Assessment Name (Eg: End/Sup)	**END**	**Paper Code**	**CJAV/G25-E**
Module Name	**Programming using JAVA**	**Module Code**	**C7-JAV-11**
Semester (E.g Jul-Dec-2021)	**Jul – Dec 2025**	**Submission Mode** (E.g Turnitin, blackboard, hardcopy)	**Turnitin & Blackboard**
Total Marks	**100**	**Duration** **(E.g., mins/hours/days/weeks**	**4 Weeks**
Assessment Type (Eg: Written/Practical/Submission)	**Submission**	**Submission Date** (Applicable for submission-based exams only)	**27th November 2025** **23:59 Hrs**

Instructions

1. No late submission allowed.
2. Students should submit the project report on turnitin and the project artefact on blackboard on or before 23:59 27th Nov 2025.
3. Database should be exported and include in the project files.
4. The database name should be the student's name and student number e.g JohnDoe12345678
5. The Project Name should follow the same naming structure.
6. Include all external Jar files in the project folders and they should be added imported into the project from there giving the paths as relative paths.
7. The database of choice is strictly MySQL.
8. Submit an executable jar file along with the project artefact on blackboard.
9. Include a readme file in your project to explain how to get started and running your project.
10. The report should have a cover page with your particulars and contents should be in an orderly manner.
11. Similarity Index should not exceed 20%.
12. Usage of AI tools in this assessment is not allowed.

University Clubs Management System: Club Connect

Project Overview:

As a Java developer consulted by the university's student affairs office, you are tasked with addressing inefficiencies in managing extracurricular clubs. Current manual methods (e.g., emails, spreadsheets) lead to issues like event scheduling conflicts, lost membership applications, and poor budget tracking, resulting in low student engagement and administrative overhead. Your goal is to develop a desktop application, ClubConnect, to streamline club operations for admins, club leaders, and members. The application must be built in Java using Java Swing for the GUI, incorporating object-oriented programming (OOP), collections, MySQL for primary data persistence (with certain data in CSV files), multi-threading, and exception handling to solve this real-world problem.

Upon running the application, it should automatically create the MySQL database and tables if they do not exist, and populate those using data from a previously exported database file (e.g., SQL dump or CSV imports). When closing the application, the database should be exported to a file for backup.

All club meetings must be registered as events, including resource bookings (e.g., rooms). Attendance should be taken during the meeting. When registering an event, include an option to request a budget (not required for all events), which must be approved by the admin.

Multi-threading should be used in the following scenarios: - For sending batch notifications or announcements to prevent UI freezing. - For automated background tasks like checking and sending reminders for upcoming events or low budgets. - For handling concurrent operations, such as polling for discussion board responses or exporting the database on app close without blocking the main thread.

User Roles

The application supports distinct user roles with specific permissions:

- **Admin (University Staff/Advisor):** Oversees all clubs, approves new clubs and budget requests, manages user accounts, resolves disputes, accesses global reports, and searches for members (displaying user details and associated clubs).
- **Club Leader (e.g., President):** Manages their club, including editing details, approving memberships, creating events (including meetings with resource bookings and optional budget requests), allocating budgets, sending notifications, and taking attendance.
- **Member (Student):** Browses/joins clubs, RSVPs to events, views announcements, submits feedback, participates in discussions, and manages personal profiles.
- **Guest (Unauthenticated):** Views public club/event listings but cannot interact until registered

User Workflows

The following outlines the key workflows for each user role, describing step-by-step interactions with the application.

1. ### Admin Workflow:

   Has overarching control. Can oversee all clubs, approve new club creations, manage user accounts (e.g., promote members to leaders), resolve disputes, access global reports, approve budget requests for events, and perform advanced searches for members (including details and associated clubs). They act as moderators to maintain compliance with university policies.

2. ### Club Leader Workflow

   Manages their specific club. Responsibilities include editing club details, approving membership applications, creating and scheduling events (including meetings/seatings with room bookings), requesting budgets for events (optional), sending notifications to members, and generating club-specific reports. They have read-only access to other clubs for inspiration or collaboration ideas.

3. ### Member Workflow

   Basic users who join clubs. They can browse and search for clubs, apply for membership, RSVP to events (including meetings/seatings), view club announcements, submit feedback, and access personal profiles or event histories. No editing rights for club-level data to maintain security.

4. ### Guest Workflow

   Limited access for prospective users. Can view public club listings and event calendars but cannot interact (e.g., no applications or RSVPs) until registering and logging in.

User roles should be enforced through authentication checks in the code, with potential for role-based interfaces (e.g., different dashboard views).

Deliverables and Assessment

The project is divided into two sections: Section A: Artifact (70 marks) and Section B: Project Report and Presentation (30 marks). The total marks are 100.

Section A: Artifact (70 marks)

This section evaluates the functional implementation and user interface of the ClubConnect application. The artifact must meet the functional requirements and provide an intuitive, professional look and feel.

Functional Requirements (60 marks)

These outline the core behaviors the application must support, emphasizing user interactions, data flows, and system logic. Each requirement should be implemented with appropriate Java features like GUI elements, data structures, and error handling for a robust user experience. Multi-threading must be used specifically for: background notifications (e.g., batch sending announcements or reminders to avoid UI blocking)

1. *User Authentication and Roles (10 marks)*
    - Provide a login screen where users enter credentials (e.g., username/password or student ID). Support registration for new users with email verification.
    - Upon login, load role-specific dashboards: Admins see all clubs; leaders see their club tools; members see joined clubs and events.
    - Implement logout functionality and session management to prevent unauthorized access. Use secure storage for credentials (hashed passwords in MySQL).
    - Handle forgotten passwords with a basic reset mechanism (e.g., generate a temporary code).

2. *Club Creation and Management:*
    - Admins and potential leaders can submit club creation requests with details like name, category, mission statement, and initial budget proposal; admins approve to activate.
    - Allow editing of club profiles (e.g., update description, add photos via file upload), deactivation, or archiving of inactive clubs.
    - Provide search and filtering for clubs (e.g., by category, size, or keywords) using collections like ArrayLists or Maps for efficient querying.
    - Display club directories with sortable lists or tables, including metrics like member count or upcoming events.

3. *Membership Handling:*
    - Members can browse clubs and submit join applications with personal info; leaders review and approve/reject applications, with optional reasons.
    - Track membership status (e.g., pending, active, alumni) and allow updates like role promotions or removals.
    - Export member lists to CSV for administrative use.
    - Prevent duplicates by validating unique student IDs and handle capacity limits if clubs have membership caps.

4. *Event Planning and Scheduling (Enhanced for All Club Seatings/Meetings):*
    - All club activities, including meetings and seatings, must be registered as events. Club leaders create events with details like title, date, time, venue/room (with automated booking check), description, capacity, and type (e.g., meeting, seating, event). Integrate a calendar view for visualization.
    - During event creation, include an optional checkbox for requesting university budget allocation (not all events require it); if selected, generate a budget request form with estimated costs, which is automatically submitted for admin approval.
    - Check for scheduling conflicts (e.g., room/venue overlaps across clubs) using algorithms on a shared event repository.
    - During the event (or post-event), leaders can mark attendance via a check-in interface.
    - Support event updates/cancellations with automatic notifications to RSVPed users.
    - Resource booking: Events requiring rooms must book available resources (e.g., query MySQL for availability); unavailability should trigger an alert and suggest alternatives.

5. *Budget and Resource Tracking (Enhanced for Event-Specific Requests):*
    - Leaders allocate and track club budgets, logging expenses/revenues. For events, if budget is requested during creation, admins must approve/reject via their dashboard, with notifications sent upon decision.
    - Admins approve budget requests and view audit trails for all transactions.
    - Include resource management for bookable items like rooms: Query availability during event creation, auto-book if free, and release upon event end.

6. *Communication and Notifications:*
    - Enable club leaders to send announcements or polls to members via in App Messages and Email Notifications, use multi-threading for batch sending to large member lists.
    - Provide a discussion board or comment section for events/clubs, with moderation tools for leaders/admins.
    - Automate reminders (through emails) for upcoming events and deadlines.

7. *Admin-Specific Search for Members:*
    - Admins can perform searches for members using criteria like name, student ID, or role. Results display full user details (e.g., contact info, profile) and all associated clubs (e.g., list of joined clubs with statuses).
    - Search results should be paginated if large, using collections for efficient querying from MySQL.

8. *Reporting and Analytics:*
    - Generate customizable reports, such as event attendance summaries (including for meetings/seatings), club engagement metrics (using maps to count), or budget overviews. Include visualizations like bar charts for participation trends via Swing components.
    - Allow exports in formats like PDF or CSV for sharing with university officials.
    - Provide analytics for admins, like overall campus club activity or popular event types, to inform decisions.

9. *Data Persistence:*
    - Use MySQL for primary storage of clubs, members, events, and budgets via JDBC; certain data (e.g., exports, logs) in CSV files.
    - Automatically create DB/tables on start if not exist; populate from exported file (e.g., SQL dump or CSV).
    - Export DB to file on app close (using multi-threading to avoid blocking); support import/export for migration.

10. *Error Handling and Validation:*
    - Validate all user inputs (e.g., date formats, email validity, room availability) and handle exceptions gracefully with user-friendly messages.
    - For event/budget flows, validate optional budget requests and ensure approvals are required before proceeding.

11. *Look and Feel*
    - Design an intuitive, professional GUI using Java Swing with consistent navigation (e.g., menus, buttons, tabs).
    - Use layouts like tables, forms, and calendars for clear data presentation (e.g., club lists, event schedules).
    - Apply basic styling for visual appeal, including fonts, colors, and spacing.
    - Provide user feedback (e.g., success/error dialogs) for all interactions.

Section B: Project Report and Presentation (30 marks)

This section evaluates the documentation and presentation of the project, ensuring clarity in design and effective communication of functionality.

Project Report (25 marks)

- **UML Diagrams:**
  Provide class and use-case diagrams showing relationships between entities (e.g., Club, Member, Event) and user interactions.
- **Javadoc Documentation:**
  Include Javadoc comments for all classes, methods, and key logic, explaining purpose and usage.
- **README and User Manual:**
  Submit a README with setup instructions (e.g., dependencies, run commands) and a user manual detailing how to use the application.
- **ERD (Entity-Relationship Diagram):**
  Provide an ERD illustrating the database schema, including tables, relationships, keys, and attributes for MySQL storage.

Presentation (5 marks)

- Deliver a presentation of about 10-15 minutes showcasing all functional requirements and the GUI.
- Highlight key features, user workflows (e.g., admin approving a club, member RSVPing), and error handling.
- Explain design choices (e.g., why specific collections or persistence methods were used).
- Ensure clear narration or text, professional delivery, and coverage of all user roles.

Marks Breakdown:

Criteria	Sub - Criteria	Marks	Description
Section A: Artifact (70 marks)			
User Authentication and Roles (10)	Login/Registration	4	Fully functional login and registration with validation and email verification.
	Role-Based Dashboards	3	Customized dashboards load correctly for Admin, Leader, and Member roles.
	Logout/Session Management	2	Secure logout with session management to prevent unauthorized access.
	Secure Storage/Reset	1	Hashed credential storage in MySQL with password reset mechanism.
Club Creation and Management (10)	Creation/Approval	3	Full workflow for club creation with admin approval process.
	Editing/Deactivation	3	Smooth editing, deactivation, or archiving of clubs.
	Filtering	2	Efficient filtering of clubs by category or size.
	Metrics Display	2	Accurate display of metrics like member count or upcoming events.
Membership Handling (8)	Browse/Apply/Approve	3	Full application workflow with leader approval and feedback.
	Status Tracking/Changes	3	Accurate tracking of membership status and

			role updates.
	Validation/Caps	2	Robust validation of student IDs, membership caps; admin-only member search with details and associated clubs.
Event Planning and Scheduling (9)	Event Creation/Calendar	3	Full event creation (including meetings) with calendar view, resource booking, optional budget request.
	Conflict Checks	2	Accurate algorithm for detecting scheduling conflicts (e.g., room bookings).
	RSVPs/Waitlists/Attendance	3	Support for RSVPs, waitlists, and attendance marking during events/meetings.
	Updates/Cancellations	1	Handle event updates/cancellations with notifications.
Budget and Resource Tracking (7)	Allocation/Logging	2	Full budget allocation and expense/revenue logging by category; integrate with event budget requests.
	Approvals/Histories	2	Admin approvals for budgets (including event-specific) and transaction history views.
	Resource Management	2	Availability checks for shared resources (e.g., rooms) during event creation.
Communication and Notifications (6)	Announcements/Polls	3	Functional in-app announcements or polls (multi-threaded batch sending).
	Discussion Boards	3	Moderated discussion boards for clubs/events.
Reporting and Analytics (7)	Engagement Reports	2	Accurate engagement reports (e.g., attendance, growth).
	Visualizations	2	Integrated charts using Swing components.
	Analytics and Exports	3	Export Attendance document with member names prepopulated as pdf. Create a financial reports of clubs
Data Persistence (5)	MySQL DB Connection & Auto-Creation	3	Robust MySQL storage with auto-creation and population.
	Export on Close/Backup	1	Automatic DB export on close (multi-threaded).
	Import/Export	1	Support for data import/export (e.g., SQL dump, CSV).
Error Handling and Validation (3)	Input Validation	1	Comprehensive input validation with immediate feedback.
	Exception Handling	1	User-friendly dialogs for exceptions (e.g., DB errors).
	Error Logging/Retries	1	Error logging and retry mechanisms for critical operations.
Look and Feel (5)	Intuitive GUI/Navigation	2	Consistent, user-friendly Swing GUI with clear navigation.
	Layouts/Presentation	1	Effective use of tables, forms, calendars for data.
	Styling/Visual Appeal	1	Professional styling with fonts, colors, spacing.
	Responsiveness/Edge Cases	1	Handles different screen sizes and large datasets.

Section B: Project Report and Presentation (30 marks)			
UML Diagrams (8)	Class Diagrams	4	Complete entities and relationships depicted accurately.
	Use-Case Diagrams	4	Detailed user interactions for all roles.
Javadoc Documentation (6)	Coverage	3	Javadoc for all classes, methods, and key logic.
	Clarity/Explanation	3	Clear explanation of purpose and usage.
README and User Manual (6)	README Setup	3	Detailed setup instructions (dependencies, run commands).
	User Manual	3	Comprehensive guide for application usage.
ERD (Entity-Relationship Diagram) (5)	Schema/Tables/Relationships	3	Complete MySQL schema with tables and relationships.
	Keys/Attributes	2	Accurate keys and attributes defined.
Presentation (5)	Demo Showcase	2	Covers all features and workflows in presentation (15 minutes).
	Explanations	2	Clear explanation of design choices.
	Delivery	1	Professional narration or text, clear delivery.
Total		**100**	