STATE UNIVERSITY OF ZANZIBAR (SUZA)

School of Computing Communication and Media Studies (SCCMS)

Group Project: Advanced Java Application

Teams of 3 Students

Course Code:	IT6003	**Programme:**	MSc. Information T
Course Name:	Advanced Programming Using Java	**Semester:**	I
Team Size:	3 students	**Total Marks:**	100

1 Project Overview

Each team of **3 students** will design, implement, and present a complete Java application that demonstrates mastery of advanced Java concepts covered in this course. The project must integrate **at least 6** of the following topics:

Core Topics (must use at least 4)	Advanced Topics (must use at least 2)
Collections Framework	Design Patterns (at least 2 patterns)
Generics	Multithreading / Concurrency
Lambda Expressions	Networking (Sockets / HTTP)
Stream API	JavaFX GUI
File I/O / NIO	JDBC / Database
Exception Handling	

2 Project Options

Each team must choose **ONE** of the following projects. No two teams may choose the same project (first-come, first-served basis). Teams may propose an alternative project of equivalent complexity, subject to instructor approval.

2.1 Option 1: Student Academic Records Management System

A desktop application for managing student records at a university department.

Features:

- **Student Management:** Add, update, delete, search students with registration number, name, programme, year of study, and photo
- **Course Management:** Add courses with code, name, credits, and prerequisites
- **Enrollment:** Enroll students in courses, check prerequisite completion, enforce maximum capacity
- **Grading:** Enter grades, calculate GPA (per semester and cumulative), generate transcripts
- **Reports:** Dean's list, academic probation list, department statistics, enrollment trends
- **Data Persistence:** SQLite database via JDBC
- **GUI:** JavaFX with dashboard, data tables, charts
- **Export:** Generate transcript as formatted text file

Required Advanced Concepts:

- Collections: HashMap for student lookup, TreeMap for sorted grade reports
- Generics: Generic DAO (Data Access Object) pattern
- Streams: GPA calculation, filtering, statistical reports

- JDBC: Full CRUD with prepared statements and transactions
- JavaFX: TableView, Charts, Form validation
- Design Patterns: Singleton (DB connection), Observer (auto-refresh), MVC architecture

2.2 Option 2: Real-Time Chat Application with File Sharing

A multi-user chat application with private messaging, group chats, and file transfer.

Features:

- **User Authentication:** Login/register with username and password (stored in database)
- **Chat Rooms:** Create, join, leave public/private chat rooms
- **Private Messaging:** Direct messages between two users
- **File Sharing:** Send files (images, documents) up to 10MB within chats
- **Online Status:** Show online/offline status of users
- **Chat History:** Save and load message history from database
- **GUI:** JavaFX with contact list, chat window, emoji support
- **Notifications:** Visual/sound notification for new messages

Required Advanced Concepts:

- Networking: TCP sockets for real-time messaging, multithreaded server
- Multithreading: Thread pool for handling clients, concurrent collections for shared state
- Collections: ConcurrentHashMap for online users, LinkedList for message queues
- File I/O: Binary file transfer over sockets, NIO for efficient file handling
- JDBC: User accounts, message history, chat room persistence
- JavaFX: Real-time UI updates using Platform.runLater()
- Design Patterns: Observer (message delivery), Factory (message types), Singleton (server instance)

2.3 Option 3: Personal Finance Manager

A desktop application for tracking income, expenses, budgets, and financial goals.

Features:

- **Accounts:** Multiple accounts (cash, bank, mobile money) with balances
- **Transactions:** Record income/expenses with category, date, amount, description
- **Categories:** Predefined and custom categories (Food, Transport, Rent, Education, etc.)
- **Budgets:** Set monthly budget limits per category, alerts when approaching/exceeding
- **Reports:** Monthly/yearly spending breakdown, income vs expenses, category analysis
- **Data Visualization:** Pie charts (spending by category), line charts (spending trends), bar charts (monthly comparison)
- **Import/Export:** Import transactions from CSV, export reports to text file
- **Search and Filter:** By date range, category, amount range, keyword

Required Advanced Concepts:

- Collections: TreeMap for date-ordered transactions, HashMap for category totals
- Generics: Generic filter/search methods
- Streams: Aggregate calculations, grouping, partitioning
- Lambdas: Custom comparators, predicates for filtering
- JDBC: SQLite for all data persistence with transactions
- JavaFX: Dashboard with charts, TableView, DatePicker, form validation
- Design Patterns: Strategy (report generation), Observer (budget alerts), Builder (transaction creation), MVC

2.4 Option 4: Library Management System with Online Catalog

A complete library management system with book catalog, member management, and borrowing system.

Features:

- **Book Catalog:** Add, update, remove books with ISBN, title, author, category, copies available

- **Member Management:** Register members, track membership status, borrowing history
- **Borrowing System:** Check out/return books, due dates, fine calculation for overdue books
- **Reservation:** Reserve books that are currently borrowed, automatic notification when available
- **Search:** Advanced search by title, author, ISBN, category with auto-complete suggestions
- **Reports:** Most borrowed books, active members, overdue books, category statistics
- **Data Persistence:** SQLite database
- **Barcode/ID:** Generate and display book ID as text-based barcode

Required Advanced Concepts:

- Collections: HashMap for book lookup, PriorityQueue for reservation waitlist, TreeSet for sorted catalogs
- Generics: Generic search/filter methods, generic Repository pattern
- Streams: Report generation, statistical analysis
- File I/O: Export catalog to CSV, import books from CSV
- JDBC: Relational database with foreign keys, joins, transactions
- JavaFX: TableView with inline editing, search with FilteredList, charts
- Design Patterns: Observer (reservation notifications), Strategy (fine calculation), Singleton (database), MVC

2.5 Option 5: Multi-Player Quiz Game

A networked quiz game where players compete in real-time.

Features:

- **Question Bank:** Multiple choice questions stored in database, organized by category and difficulty
- **Game Modes:** Single player (vs timer), multiplayer (2–4 players over network)
- **Networking:** Server hosts the game, clients join via IP/port
- **Real-Time:** All players see the same question simultaneously, timer counts down
- **Scoring:** Points based on correctness and speed, live leaderboard
- **Game Flow:** Lobby → Ready check → Rounds → Results → Leaderboard
- **Question Editor:** Admin interface to add/edit/delete questions
- **Statistics:** Player history, win/loss record, category performance

Required Advanced Concepts:

- Networking: TCP server-client architecture, protocol design for game messages
- Multithreading: Game timer, concurrent player handling, synchronized game state
- Collections: ConcurrentHashMap for player sessions, ArrayList for questions
- Generics: Generic message handling for different message types
- JDBC: Question bank, player statistics, game history
- JavaFX: Game UI with animations, countdown timer, live score updates
- Design Patterns: Observer (score updates), Factory (question types), State (game phases), Singleton (game server)

3 Team Responsibilities

Each team member must have a clearly defined role. All members must contribute to coding, but primary responsibilities should be divided as follows:

Role	Title	Responsibilities
Member 1	**Lead Developer / Architect**	System design, core architecture, design patterns, code integration, code review
Member 2	**Backend Developer**	Database design, JDBC implementation, business logic, file I/O, networking (if applicable)
Member 3	**Frontend Developer / QA**	JavaFX GUI, CSS styling, user experience, testing, documentation

Important: Each member must be able to explain **any** part of the code, not just their own. The viva will test understanding of the entire project.

4 Deliverables

4.1 Submission Package

Submit a ZIP file named `TeamName_Project.zip` containing:

1. **Source Code:** All `.java` files, `.fxml` files, `.css` files, organized in proper package structure
2. **Database:** SQLite database file with sample data (at least 20 records per table)
3. **Documentation:** (see Section 4.2)
4. **README.txt:** Compilation and running instructions, dependencies, known issues
5. **Presentation:** PowerPoint/PDF slides for the project demo

4.2 Documentation

A written report (5–8 pages, PDF format) containing:

1. **Title Page:** Project name, team members (name, reg no, role), date
2. **Introduction:** Problem statement, objectives, scope
3. **System Design:**
   - Class diagram (UML) showing all classes and relationships
   - Database schema (ER diagram or table definitions)
   - Architecture diagram showing how components interact
4. **Advanced Java Concepts Used:** For each concept, explain:
   - Where it is used in the code (class name, method)
   - Why it was chosen over alternatives
   - A brief code snippet demonstrating its use
5. **Design Patterns:** For each pattern used:
   - Pattern name and intent
   - Class diagram of the pattern in your project
   - Explanation of why this pattern was appropriate
6. **Challenges and Solutions:** Technical challenges faced and how they were resolved
7. **Team Contribution:** Table showing each member's contribution (be specific and honest)
8. **Conclusion:** What was learned, potential improvements

5 Presentation and Viva

5.1 Project Presentation (15 minutes per team)

- 5 minutes: System overview and architecture (slides)
- 7 minutes: Live demonstration of the application
- 3 minutes: Q&A from the instructor and class

5.2 Individual Viva (5 minutes per member)

Each team member will be asked individually to:

1. Explain the overall system architecture
2. Walk through a specific class they wrote
3. Explain a design pattern used in the project
4. Modify a small feature live (e.g., add a new filter, change a validation rule)
5. Answer conceptual questions about the advanced Java topics used

Warning: If a team member cannot explain their contribution or the overall system, their individual score will be **zero** regardless of the team's project quality.

6 Grading Rubric

Component	Marks	Details
Functionality	30	All features work correctly, handles edge cases, no crashes
Code Quality	20	Clean, well-organized code; proper naming conventions; meaningful comments; proper package structure; no code smells
Advanced Concepts	20	Correct and meaningful use of at least 6 advanced Java concepts; concepts are integral to the solution (not added superficially)
Design Patterns	10	At least 2 design patterns correctly implemented and justified
Documentation	10	UML diagrams, report quality, README completeness
Presentation & Demo	10	Clear presentation, working demo, handles questions well
Total	**100**	

Individual Multiplier: Each member receives a multiplier (0.0 to 1.0) based on their viva performance. Final individual mark = Team mark × Individual multiplier.

Bonus Points (up to 10 extra marks):

- Exceptional UI design and user experience (+3)
- Use of additional advanced concepts beyond the minimum (+3)
- Comprehensive testing with test cases documented (+2)
- Creative and original feature additions (+2)

7 Academic Integrity

- Each team's code must be original. Sharing code between teams is **strictly prohibited**.
- Using AI tools (ChatGPT, GitHub Copilot) is allowed as a *learning aid*, but all team members must fully understand every line of code.
- Code plagiarism detection tools will be used to compare submissions between teams.
- External libraries are **not allowed** unless approved by the instructor. Use only standard Java SE libraries (java.*, javax.*, javafx.*).
- Attribution: If you use code snippets from tutorials or documentation, cite the source in comments.

8 Timeline

Milestone	Deliverable	Weight
Week 1	Team formation, project selection, initial design	—
Week 2	Project proposal (1-page: objectives, features, team roles)	5%
Week 4	Progress check: core functionality demo	10%
Week 6	Final submission + presentation + viva	85%

End of Group Project Brief