ClubConnect - University Clubs Management System

A comprehensive Java Swing desktop application for managing university extracurricular clubs, events, memberships, and budgets.

Project Overview

ClubConnect streamlines club operations for admins, club leaders, and members by replacing manual methods (emails, spreadsheets) with an integrated desktop application. The system handles club creation, membership management, event scheduling with resource booking, budget tracking, and communication through a user-friendly GUI.

Features

User Roles

- **Admin**: Oversees all clubs, approves club creations, manages users, approves budgets
- **Club Leader**: Manages their club, creates events, approves memberships, tracks budgets
- **Member**: Browses clubs, joins clubs, RSVPs to events, views announcements
- **Guest**: Views public clubs and events (read-only)

Core Functionality

- ✅ User authentication with role-based access control
- ✅ Club creation, management, and approval workflow
- ✅ Event planning with resource booking and conflict detection
- ✅ Budget requests and approval system
- ✅ Membership application and approval process
- ✅ Multi-threaded notifications and announcements
- ✅ Attendance tracking for events
- ✅ Discussion boards for clubs and events
- ✅ Comprehensive reporting (attendance, budget, engagement)
- ✅ CSV and PDF export capabilities
- ✅ Automated database backup on application close

Prerequisites

Software Requirements

- Java JDK 11 or higher
- MySQL Server 8.0 or higher
- IDE: IntelliJ IDEA, Eclipse, or VS Code with Java extensions

Required External Libraries

1. **MySQL Connector/J** (JDBC Driver)
   - Version: 8.0.33 or higher
   - Download: https://dev.mysql.com/downloads/connector/j/
2. **iText** (PDF Generation)
   - Version: 5.5.13.3 (free version)
   - Download: https://mvnrepository.com/artifact/com.itextpdf/itextpdf/5.5.13.3
3. **JavaMail API** (Email Notifications)
   - Version: 1.6.2
   - Download: https://mvnrepository.com/artifact/javax.mail/mail/1.4.7
   - Also requires: activation.jar

Installation & Setup

1. Database Setup

```sql
-- Create MySQL database user (if needed)
CREATE USER 'clubconnect'@'localhost' IDENTIFIED BY 'your_password';
GRANT ALL PRIVILEGES ON *.* TO 'clubconnect'@'localhost';
FLUSH PRIVILEGES;

-- The application will automatically create the database and tables
```

2. Configure Database Connection

Edit `src/com/clubconnect/database/DatabaseManager.java`:

```java
```

```
private static final String DB_USER = "root"; // Change to your MySQL username
private static final String DB_PASSWORD = ""; // Change to your MySQL password
```

3. Add External Libraries

Option A: Using IDE (IntelliJ IDEA)

1. File → Project Structure → Libraries
2. Click "+" → Java
3. Select downloaded JAR files (mysql-connector, itext, mail)
4. Click OK

Option B: Using Eclipse

1. Right-click project → Build Path → Configure Build Path
2. Libraries tab → Add External JARs
3. Select JAR files → Apply and Close

Option C: Using Maven (pom.xml)

xml

```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.33</version>
    </dependency>
    <dependency>
        <groupId>com.itextpdf</groupId>
        <artifactId>itextpdf</artifactId>
        <version>5.5.13.3</version>
    </dependency>
    <dependency>
        <groupId>javax.mail</groupId>
        <artifactId>mail</artifactId>
        <version>1.4.7</version>
    </dependency>
</dependencies>
```

4. Project Structure

Create the following directory structure:

```
ClubConnect/
├── src/
│   └── com/
│       └── clubconnect/
│           ├── ClubConnectApp.java
│           ├── database/
│           ├── models/
│           ├── dao/
│           ├── services/
│           ├── ui/
│           │   ├── panels/
│           │   └── dialogs/
│           └── utils/
├── lib/ (place JAR files here)
├── resources/
└── database_backup.sql (auto-generated)
```

5. Initial Admin User

After first run, insert an admin user directly into MySQL:

```sql
USE clubconnect_db;

-- Password is hashed "admin123"
INSERT INTO users (student_id, username, email, password, role, status)
VALUES ('ADMIN001', 'Admin User', 'admin@university.edu',
        'SHA256_HASH_OF_admin123', 'ADMIN', 'ACTIVE');
```

Or create through registration and manually update role in database:

```sql
UPDATE users SET role = 'ADMIN' WHERE student_id = 'YOUR_STUDENT_ID';
```

Running the Application

From IDE

1. Open project in your IDE
2. Locate `ClubConnectApp.java` in `src/com/clubconnect/`

3. Right-click → Run 'ClubConnectApp.main()'

From Command Line

```bash
# Compile
javac -cp ".:lib/*" -d bin src/com/clubconnect/**/*.java

# Run
java -cp "bin:lib/*" com.clubconnect.ClubConnectApp
```

Create Executable JAR

```bash
# Create manifest file
echo "Main-Class: com.clubconnect.ClubConnectApp" > manifest.txt
echo "Class-Path: lib/mysql-connector-java-8.0.33.jar lib/itextpdf-5.5.13.3.jar lib/mail-1.4.7.jar" >> manifest.txt

# Create JAR
jar cfm ClubConnect.jar manifest.txt -C bin . lib/

# Run JAR
java -jar ClubConnect.jar
```

User Guide

First Time Setup

1. Launch the application
2. Database and tables are automatically created
3. Click "Register" to create an account
4. Login with your credentials

For Members

1. **Browse Clubs**: Use search and filters to find clubs
2. **Join Club**: Click "Join" button on club listings
3. **View Events**: Check upcoming events in Events tab
4. **RSVP**: Click RSVP button for events you want to attend

5. **Update Profile**: Edit your information in Profile tab

For Club Leaders

1. **Create Club**: Submit club creation request (requires admin approval)
2. **Manage Members**: Approve/reject membership applications
3. **Create Events**: Schedule meetings and events with room bookings
4. **Request Budget**: Optional budget requests during event creation
5. **Send Announcements**: Notify all club members
6. **Track Attendance**: Mark attendance during events

For Admins

1. **Approve Clubs**: Review and approve new club requests
2. **Manage Users**: Search users, view details, promote to leaders
3. **Approve Budgets**: Review and approve budget requests
4. **View Reports**: Access global analytics and reports
5. **Resolve Issues**: Handle disputes and violations

Features Documentation

Event Resource Booking

- Automatically checks room availability
- Prevents double-booking conflicts
- Suggests alternatives if resource unavailable

Budget Management

- Track club finances by category
- Event-specific budget requests
- Admin approval workflow
- Transaction history and audit trail

Notifications System

- Multi-threaded batch sending
- In-app notifications

- Email notifications (configured)
- Event reminders (automated)

Attendance Tracking

- Pre-populated with RSVPs
- Check-in interface during events
- Attendance reports with member names
- Export to PDF

Reporting & Analytics

- Attendance summaries
- Club engagement metrics
- Budget overviews
- Bar chart visualizations
- Export to PDF/CSV

Troubleshooting

Common Issues

"Cannot connect to database"

- Ensure MySQL server is running
- Check username/password in DatabaseManager.java
- Verify MySQL port (default 3306) is not blocked

"ClassNotFoundException: com.mysql.cj.jdbc.Driver"

- MySQL Connector JAR not in classpath
- Add JAR to project libraries

"Access denied for user"

- Check MySQL user permissions
- Grant appropriate privileges

UI not displaying properly

- Verify Swing EDT thread usage
- Check all components are added to containers
- Ensure setVisible(true) is called

Multi-threading issues

- Use SwingUtilities.invokeLater() for UI updates
- Proper exception handling in threads
- Don't block main UI thread

Debug Mode

Enable detailed logging:

```java
// In main method
System.setProperty("java.util.logging.config.file", "logging.properties");
```

Database Schema

Key tables:

- `users` - User accounts and authentication
- `clubs` - Club information and status
- `club_members` - Membership relationships
- `events` - Event scheduling and details
- `rsvps` - Event attendance tracking
- `resources` - Bookable resources (rooms, equipment)
- `resource_bookings` - Resource reservations
- `budget_transactions` - Financial tracking
- `announcements` - Club communications
- `discussions` - Discussion boards
- `notifications` - User notifications

See DATABASE_SCHEMA.md for complete ERD and relationships.

Project Structure

- **models/** - Entity classes (POJOs)
- **dao/** - Data Access Objects (database operations)
- **services/** - Business logic layer
- **ui/** - Swing GUI components
  - **panels/** - Reusable UI panels
  - **dialogs/** - Popup dialog windows
- **utils/** - Helper utilities
- **database/** - Database management

Development Notes

Design Patterns Used

- DAO Pattern for data access
- Singleton for SessionManager
- MVC pattern for UI separation
- Observer pattern for notifications

Multi-Threading Implementation

- NotificationService: Batch email sending
- DatabaseManager: Background database export
- EventService: Automated event reminders
- DiscussionBoard: Message polling

Security Features

- Password hashing (SHA-256)
- Role-based access control
- SQL injection prevention (PreparedStatements)
- Session management

Testing

Unit Testing Template

```java
public static void main(String[] args) {
    DatabaseManager.initializeDatabase();

    // Test your functionality
    UserDAO userDAO = new UserDAO();
    User user = new User("TEST001", "Test User", "test@test.com", "password");
    boolean created = userDAO.createUser(user);
    System.out.println("User created: " + created);

    DatabaseManager.closeConnection();
}
```

Integration Testing Checklist

- [ ] User registration and login
- [ ] Club creation and approval
- [ ] Event creation with resource booking
- [ ] Membership application workflow
- [ ] Budget request and approval
- [ ] Attendance tracking
- [ ] Report generation
- [ ] Notification sending
- [ ] Database export on close

Contributing

This is an academic project. Contributions should follow:

1. Code with proper Javadoc comments
2. Follow existing naming conventions
3. Test thoroughly before committing
4. Update documentation as needed

License

Support

For issues or questions:

- Check troubleshooting section above
- Review implementation guide
- Refer to project requirements document
- Consult instructor/TA for academic guidance

Version History

- **v1.0** (Initial Release)
  - Core authentication system
  - Club and event management
  - Basic UI framework
  - Database integration

Acknowledgments

- University Student Affairs Office (project sponsor)
- Java Swing Documentation
- MySQL Documentation
- iText PDF Library
- JavaMail API

Project: ClubConnect
Course: Programming using JAVA (C7-JAV-11)
Semester: Jul-Dec 2025
Institution: Faculty of Engineering and Technology