ClubConnect - Project Structure

Directory Structure

```
ClubConnect/
├── src/
│   └── com/
│       └── clubconnect/
│           ├── ClubConnectApp.java (Main entry point)
│           ├── database/
│           │   └── DatabaseManager.java
│           ├── models/
│           │   ├── User.java
│           │   ├── Club.java
│           │   ├── Event.java
│           │   ├── ClubMember.java
│           │   ├── RSVP.java
│           │   ├── Resource.java
│           │   ├── BudgetTransaction.java
│           │   ├── Announcement.java
│           │   ├── Discussion.java
│           │   └── Notification.java
│           ├── dao/
│           │   ├── UserDAO.java
│           │   ├── ClubDAO.java
│           │   ├── EventDAO.java
│           │   ├── ClubMemberDAO.java
│           │   ├── RSVPDAO.java
│           │   ├── ResourceDAO.java
│           │   ├── BudgetTransactionDAO.java
│           │   ├── AnnouncementDAO.java
│           │   ├── DiscussionDAO.java
│           │   └── NotificationDAO.java
│           ├── services/
│           │   ├── AuthenticationService.java
│           │   ├── ClubService.java
│           │   ├── EventService.java
│           │   ├── MembershipService.java
│           │   ├── BudgetService.java
│           │   ├── NotificationService.java
│           │   └── ReportService.java
│           ├── ui/
```

```
│   │   ├── LoginFrame.java
│   │   ├── RegistrationFrame.java
│   │   ├── AdminDashboard.java
│   │   ├── LeaderDashboard.java
│   │   ├── MemberDashboard.java
│   │   ├── GuestDashboard.java
│   │   ├── panels/
│   │   │   ├── ClubListPanel.java
│   │   │   ├── EventCalendarPanel.java
│   │   │   ├── MembershipPanel.java
│   │   │   ├── BudgetPanel.java
│   │   │   ├── AnnouncementPanel.java
│   │   │   └── ReportPanel.java
│   │   └── dialogs/
│   │       ├── CreateClubDialog.java
│   │       ├── CreateEventDialog.java
│   │       ├── AttendanceDialog.java
│   │       ├── BudgetRequestDialog.java
│   │       └── MemberSearchDialog.java
│   └── utils/
│       ├── SessionManager.java
│       ├── PasswordHasher.java
│       ├── Validator.java
│       ├── EmailSender.java
│       ├── CSVExporter.java
│       └── PDFGenerator.java
├── lib/
│   ├── mysql-connector-java-8.0.x.jar
│   ├── itext-x.x.x.jar (for PDF generation)
│   └── javax.mail.jar (for email sending)
├── resources/
│   ├── icons/
│   └── images/
├── database_backup.sql
├── README.md
└── pom.xml (if using Maven)
```

Required External Libraries

1. **MySQL Connector/J** - For database connectivity
   - Download: https://dev.mysql.com/downloads/connector/j/
   - Add to project build path

2. **iText** - For PDF generation
    - Download: https://itextpdf.com/
    - Version 5.5.13 recommended for free use
3. **JavaMail API** - For email notifications
    - Download: https://javaee.github.io/javamail/
    - Requires activation.jar as well

Files Provided So Far

Core Application Files

- ✅ `ClubConnectApp.java` - Main entry point
- ✅ `DatabaseManager.java` - Database initialization and connection
- ✅ `LoginFrame.java` - Login interface
- ✅ `RegistrationFrame.java` - User registration

Model Classes

- ✅ `User.java` - User entity
- ✅ `Club.java` - Club entity
- ✅ `Event.java` - Event entity

DAO Classes

- ✅ `UserDAO.java` - User data access

Utility Classes

- ✅ `SessionManager.java` - Session management
- ✅ `PasswordHasher.java` - Password security
- ✅ `Validator.java` - Input validation

Files You Need to Create in Cursor

1. Remaining Model Classes

- `ClubMember.java`
- `RSVP.java`
- `Resource.java`

- `BudgetTransaction.java`
- `Announcement.java`
- `Discussion.java`
- `Notification.java`

2. Remaining DAO Classes

- `ClubDAO.java`
- `EventDAO.java`
- `ClubMemberDAO.java`
- `RSVPDAO.java`
- `ResourceDAO.java`
- `BudgetTransactionDAO.java`
- `AnnouncementDAO.java`
- `DiscussionDAO.java`
- `NotificationDAO.java`

3. Service Layer Classes

- `AuthenticationService.java`
- `ClubService.java`
- `EventService.java`
- `MembershipService.java`
- `BudgetService.java`
- `NotificationService.java` (with multi-threading)
- `ReportService.java`

4. Dashboard Classes

- `AdminDashboard.java`
- `LeaderDashboard.java`
- `MemberDashboard.java`
- `GuestDashboard.java`

5. Panel Classes (for dashboard sections)

- `ClubListPanel.java`
- `EventCalendarPanel.java`
- `MembershipPanel.java`
- `BudgetPanel.java`
- `AnnouncementPanel.java`
- `ReportPanel.java`

6. Dialog Classes (popup windows)

- `CreateClubDialog.java`
- `CreateEventDialog.java`
- `AttendanceDialog.java`
- `BudgetRequestDialog.java`
- `MemberSearchDialog.java`

7. Additional Utility Classes

- `EmailSender.java` (for email notifications)
- `CSVExporter.java` (for CSV export)
- `PDFGenerator.java` (for PDF reports)

Database Setup

MySQL Requirements

1. Install MySQL Server (8.0 or higher)
2. Create a user with appropriate permissions
3. Update credentials in `DatabaseManager.java`:

```java
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "your_password";
```

Auto-Creation

The database and tables will be created automatically when you run the application.

Multi-Threading Requirements

Implement multi-threading in:

1. **NotificationService** - Batch sending of notifications
2. **DatabaseManager** - Database export on application close
3. **EventService** - Event reminders background task
4. **DiscussionBoard** - Polling for new messages

Example:

```java
Thread notificationThread = new Thread(() -> {
    // Send batch notifications
});
notificationThread.start();
```

Next Steps in Cursor

1. **Create remaining Model classes** - Similar pattern to User, Club, Event
2. **Create DAO classes** - Follow UserDAO pattern for CRUD operations
3. **Create Service layer** - Business logic and multi-threading
4. **Create Dashboard UIs** - One for each user role
5. **Create Panels** - Reusable UI components for dashboards
6. **Create Dialog windows** - For specific actions (create club, event, etc.)
7. **Implement utilities** - Email, CSV, PDF generation
8. **Test each component** - Ensure database connections work
9. **Add error handling** - Try-catch blocks throughout
10. **Polish UI** - Consistent styling and user feedback

Key Implementation Tips

For DAO Classes

- Follow the same pattern as `UserDAO.java`
- Use PreparedStatements to prevent SQL injection

- Always close resources in finally blocks or use try-with-resources
- Handle SQLExceptions gracefully

For Service Classes

- Encapsulate business logic
- Use DAO classes for data access
- Implement validation before database operations
- Use multi-threading where specified

For UI Classes

- Extend JFrame for main windows
- Extend JPanel for reusable components
- Extend JDialog for popup windows
- Use GridBagLayout or BorderLayout for flexibility
- Provide user feedback with JOptionPane

For Multi-Threading

- Use Thread or ExecutorService
- Avoid blocking the UI thread
- Handle exceptions in threads
- Use SwingUtilities.invokeLater() for UI updates from threads

Testing Checklist

- [ ] Database connection and initialization
- [ ] User registration and login
- [ ] Club creation and approval
- [ ] Event creation with resource booking
- [ ] Budget request workflow
- [ ] Membership management
- [ ] Notifications and announcements
- [ ] Attendance tracking
- [ ] Report generation
- [ ] CSV/PDF exports
- [ ] Multi-threading operations

- [ ] Error handling
- [ ] Input validation

Common Issues and Solutions

MySQL Connection Issues

- Ensure MySQL server is running
- Check username/password in DatabaseManager
- Verify MySQL Connector JAR is in classpath

UI Not Displaying

- Check Swing EDT thread usage
- Verify setVisible(true) is called
- Check layout managers are properly configured

Multi-Threading Issues

- Use SwingUtilities.invokeLater() for UI updates
- Handle thread exceptions properly
- Don't block the main UI thread

Contact for Help

Review the project requirements document for detailed specifications of each feature.