TRƯỜNG ĐẠI HỌC CÔNG NGHỆ – ĐHQGHN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CÁ NHÂN

Sử dụng công cụ hợp tác trực tuyến trong dự án nhóm

Họ và tên:	Lê Anh Quang
Mã sinh viên:	25021951
Dự án nhóm:	Xây dựng ứng dụng đấu giá trực tuyến (Java Desktop App)

I. GIỚI THIỆU VÀ BỐI CẢNH DỰ ÁN

1.1. Tổng quan dự án

Nhóm chúng tôi gồm 4 thành viên triển khai dự án xây dựng ứng dụng đấu giá trực tuyến (Auction App) sử dụng ngôn ngữ Java. Ứng dụng cho phép người dùng đăng ký tài khoản, đăng sản phẩm lên đấu giá, đặt giá thầu theo thời gian thực và theo dõi lịch sử giao dịch. Kiến trúc được thiết kế theo mô hình Client–Server: phía server dùng Java Socket + Thread Pool xử lý đồng thời nhiều phiên đấu giá; phía client là giao diện desktop Java Swing. Dữ liệu được lưu trữ trên MySQL thông qua JDBC.

Dự án chia thành 3 mảng: Core Auction Engine (xử lý logic đặt giá, đồng bộ real-time), GUI & UX (giao diện Swing), và Data Layer (thiết kế schema, JDBC). Cá nhân tôi đảm nhận vai trò kỹ sư Core Auction Engine kiêm Trưởng nhóm kỹ thuật, chịu trách nhiệm điều phối tiến độ, thiết kế giao thức Client-Server, và đảm bảo tính nhất quán dữ liệu khi nhiều người đặt giá đồng thời.

1.2. Lý do lựa chọn công cụ

Với đặc thù dự án Java có nhiều thành phần kỹ thuật phức tạp và 4 thành viên làm việc độc lập, tôi xác định bốn nhu cầu cốt lõi: (1) phân công và theo dõi task kỹ thuật chi tiết theo từng module, (2) quản lý phiên bản mã nguồn và review code có hệ thống, (3) soạn thảo tài liệu thiết kế đồng thời, và (4) trao đổi kỹ thuật có thể tra cứu lại. Trên cơ sở đó, tôi lựa chọn bộ 5 công cụ: Trello, GitHub, Google Docs, Google Drive và Discord, bao phủ đầy đủ 4 nhóm chức năng theo yêu cầu bài tập và bổ sung thêm lớp quản lý mã nguồn chuyên dụng thiết yếu cho dự án lập trình.

II. THIẾT LẬP VÀ SỬ DỤNG CÔNG CỤ

2.1. Trello – Quản lý dự án

Tôi tạo board "AuctionApp-Java" với 5 list theo quy trình Kanban: Backlog, To Do, In Progress, Review/Test, Done. Để đạt hiệu quả tối ưu (Mức 4), tôi áp dụng các kỹ thuật nâng cao:

- Labels màu theo module: đỏ (Auction Engine), xanh lam (GUI/Swing), vàng (Data Layer/JDBC), xám (DevOps/Build), tím (Documentation).
- Power-Up Butler tự động hóa: khi card được chuyển sang Done, Butler tự gắn nhãn DONE-[ngày] và gửi webhook thông báo vào kênh #trello-updates trên Discord.
- Custom Fields trên mỗi card: Estimated Hours, Actual Hours, Priority (P1/P2/P3), GitHub PR Link, và trạng thái Unit Test (Pass/Fail/Pending).
- Template card chuẩn cho mọi task kỹ thuật: checklist 6 bước – Phân tích yêu cầu, Thiết kế class/interface, Implement, Viết unit test, Code review trên GitHub, Merge.

Tổng cộng tôi quản lý 14 card trong tuần, cập nhật tiến độ trung bình 2 lần/ngày. Mỗi card được liên kết trực tiếp với Pull Request tương ứng trên GitHub thông qua Custom Field GitHub PR Link, tạo ra sự liên thông giữa task quản lý và thay đổi thực tế trong codebase.

2.2. GitHub – Quản lý mã nguồn và review code

Tôi khởi tạo repository "auction-app-java" trên GitHub với cấu trúc nhánh theo mô hình Git Flow:

- Nhánh main: chỉ nhận merge từ develop sau khi toàn nhóm kiểm thử xong, được bảo vệ bằng Branch Protection Rule – yêu cầu ít nhất 1 approval trước khi merge.
- Nhánh develop: nhánh tích hợp chính, nơi các feature branch được merge vào sau khi pass code review.
- Nhánh feature/[tên-tính-năng]: mỗi thành viên làm việc trên nhánh riêng, ví dụ feature/auction-engine, feature/bid-manager, feature/gui-main-window.

Trong 7 ngày, cá nhân tôi:

- Tạo và merge 6 Pull Request, tổng cộng 47 commit với message theo chuẩn Conventional Commits: feat(engine): add synchronized bid validation, fix(engine): resolve race condition in BidManager.
- Review và để lại 23 comment trên 4 Pull Request của thành viên khác, tập trung vào tính đúng đắn của logic đồng bộ và tuân thủ chuẩn code style.
- Thiết lập GitHub Actions workflow tự động chạy JUnit 5 test suite mỗi khi có push lên develop, ngăn chặn code lỗi được merge vào nhánh tích hợp.
- Tạo 3 Issue để theo dõi bug phát sinh trong quá trình kiểm thử, liên kết với Trello card tương ứng qua GitHub PR Link.

Nhờ cấu trúc Git Flow rõ ràng và Branch Protection, không xảy ra tình trạng ghi đè code của nhau trong suốt tuần làm việc.

2.3. Google Docs – Soạn thảo cộng tác

Nhóm sử dụng 3 tài liệu chính: Tài liệu Thiết kế Hệ thống (System Design Document – SDD), Đặc tả giao thức Client-Server (Protocol Specification), và Biên bản họp nhóm (Meeting Notes). Cá nhân tôi đóng góp:

- 41 lần chỉnh sửa trực tiếp trong 7 ngày, ghi rõ trong Version History với tên tài khoản cá nhân.
- Sử dụng Suggestion Mode để đề xuất thay đổi mà không ghi đè phần thành viên khác đang soạn, giữ nguyên lịch sử phê duyệt.
- Dùng Comment để hỏi và giải thích ngay trên nội dung tài liệu, tránh mất ngữ cảnh khi phải giải thích lại trên Discord.
- Thiết lập Notification Rules: nhận thông báo email khi có người comment vào phần SDD do tôi phụ trách.

2.4. Google Drive – Lưu trữ và chia sẻ tệp

Tôi thiết kế hệ thống thư mục 3 cấp trên Google Drive:

- Cấp 1: AuctionApp-Java/ (Shared Drive toàn nhóm)
- Cấp 2: 01_Design/, 02_AuctionEngine/, 03_GUI/, 04_DataLayer/, 05_Build/, 06_MeetingNotes/
- Cấp 3 (ví dụ trong 02_AuctionEngine/): ClassDiagrams/, SequenceDiagrams/, UnitTests/, Refactoring_Notes/

Quy tắc đặt tên tệp nhất quán: [YYYYMMDD]_[module]_[mô-tả-ngắn]_v[phiên-bản].ext – ví dụ: 20250410_Engine_AuctionSession-sequence-diagram_v1.png. Toàn bộ 26 tệp do tôi upload đều tuân thủ 100%. Phân quyền theo vai trò: Viewer cho người ngoài nhóm, Commenter cho thành viên xem chéo, Editor chỉ cho người phụ trách module. Lưu ý: Drive được dùng để lưu tài liệu thiết kế và artifact (diagram, báo cáo test), còn mã nguồn được quản lý hoàn toàn trên GitHub để tận dụng diff và branch.

2.5. Discord – Giao tiếp nhóm

Server Discord nhóm cấu trúc 7 kênh chuyên biệt: #general, #auction-engine, #gui-swing, #data-layer, #daily-standup, #github-notifications, #resources. Trong 7 ngày, cá nhân tôi:

- Gửi 52 tin nhắn tổng cộng, bao gồm báo cáo tiến độ hàng ngày trên #daily-standup và ít nhất 3 tin nhắn kỹ thuật trong #auction-engine mỗi ngày.
- Chủ động hỗ trợ 3 thành viên xử lý lỗi kỹ thuật qua voice channel và thread riêng, bao gồm lỗi race condition trên BidManager và lỗi JDBC connection pool.
- Cấu hình GitHub webhook vào kênh #github-notifications: mọi push, Pull Request và comment review đều được đẩy thông báo tự động vào Discord, giúp cả nhóm theo dõi hoạt động code mà không cần vào GitHub liên tục.
- Cấu hình bot MEE6 nhắc lịch stand-up mỗi sáng 8h30 và Trello webhook thông báo khi card thay đổi trạng thái.

III. NHIỆM VỤ CÁ NHÂN VÀ KẾT QUẢ THỰC HIỆN

Bảng tổng hợp các nhiệm vụ cá nhân trong tuần dự án:

STT	Tên nhiệm vụ	Deadline	Hoàn thành	GitHub PR	Trạng thái
1	Thiết kế class diagram toàn hệ thống	09/04	09/04	–	Hoàn thành
2	Implement AuctionSession & BidManager	10/04	10/04	PR #2	Hoàn thành
3	Implement Server Socket + ThreadPool	11/04	11/04	PR #3	Hoàn thành
4	Viết Protocol Specification (Google Docs)	11/04	11/04	–	Hoàn thành
5	Xử lý race condition – synchronized lock	12/04	12/04	PR #5	Hoàn thành
6	Viết unit test AuctionSession (JUnit 5)	12/04	13/04	PR #6	Hoàn thành
7	Tích hợp & kiểm thử end-to-end	13/04	13/04	PR #8	Hoàn thành
8	Soạn System Design Document	14/04	14/04	–	Hoàn thành

Tất cả 8 nhiệm vụ đều được mô tả đầy đủ tên, deadline, người thực hiện trên Trello và liên kết với Pull Request tương ứng trên GitHub. Tiến độ cập nhật ít nhất 4 lần/ngày: khi bắt đầu task, khi mở PR, khi PR được approve, và khi merge hoàn tất. Nhờ đó toàn nhóm nắm trạng thái theo thời gian thực mà không cần hỏi trực tiếp.

IV. PHÂN TÍCH HIỆU QUẢ CÔNG CỤ

4.1. Trello

Ưu điểm: Kanban trực quan, Butler tự động hóa thao tác lặp. Custom Field GitHub PR Link tạo cầu nối trực tiếp giữa task quản lý và thay đổi mã nguồn thực tế, giúp PM và reviewer không cần chuyển đổi qua lại giữa hai công cụ.

Nhược điểm: Phiên bản miễn phí giới hạn Power-Up, không thể tích hợp đồng thời cả GitHub, Drive và Discord tự động hoàn toàn. Không có time tracking tích hợp, phải ghi Actual Hours thủ công.

4.2. GitHub

Ưu điểm: Quản lý phiên bản mã nguồn phân tán, Branch Protection ngăn code lỗi merge vào nhánh chính. Pull Request là không gian review code có cấu trúc: diff trực quan, comment inline theo từng dòng, CI/CD tự động chạy test. GitHub Actions tích hợp JUnit 5 giúp phát hiện regression ngay khi push, tiết kiệm thời gian kiểm thử thủ công đáng kể.

Nhược điểm: Đường cong học tập với Git Flow khá dốc cho thành viên chưa quen; hai ngày đầu mất thời gian hướng dẫn nhóm về cách tạo branch, rebase và resolve conflict. Giao diện Pull Request chưa

hỗ trợ tốt cho thảo luận dài, nên các quyết định thiết kế lớn vẫn cần được tổng hợp lại trên Google Docs.

4.3. Google Docs

Ưu điểm: Cộng tác đồng thời không xung đột, Version History chi tiết giúp truy vết thay đổi đặc tả giao thức. Kết hợp hiệu quả với GitHub: code được review trên PR, còn quyết định thiết kế đằng sau được giải thích trong Docs.

Nhược điểm: Không hỗ trợ tốt code block Java dài, cần add-on Code Blocks. Với tài liệu nặng như sequence diagram, Google Docs tỏ ra hạn chế so với draw.io.

4.4. Google Drive

Ưu điểm: Lưu trữ artifact (diagram, báo cáo test, file xuất từ IDE) tập trung, tìm kiếm toàn văn trong PDF. Phân quyền linh hoạt theo từng thư mục. Phân chia rõ vai trò với GitHub: Drive lưu tài liệu, GitHub lưu code – tránh nhầm lẫn.

Nhược điểm: Không phải môi trường quản lý phiên bản mã nguồn, không có diff hay branch. Nếu team không phân chia rõ vai trò giữa Drive và GitHub, dễ dẫn đến trùng lặp hoặc lưu sai chỗ.

4.5. Discord

Ưu điểm: Kênh #github-notifications đẩy thông báo tự động từ GitHub (push, PR, review) giúp cả nhóm theo dõi hoạt động code mà không cần kiểm tra trực tiếp. Thread riêng theo từng bug giữ thảo luận kỹ thuật gọn gàng.

Nhược điểm: Không phải môi trường chuyên nghiệp, phản hồi đôi khi chậm. Chat không có cấu trúc nên quyết định kỹ thuật quan trọng phải được tổng hợp lại trên Google Docs để tránh thất lạc.

V. BA THÁCH THỨC VÀ GIẢI PHÁP

Thách thức 1: Race condition khi nhiều client đặt giá đồng thời

Trong quá trình kiểm thử, phát hiện trường hợp hai client gửi bid cùng lúc cho một AuctionSession, dẫn đến cả hai đều được ghi nhận là winner do điều kiện kiểm tra giá cao nhất không có cơ chế khóa. Lỗi không xảy ra khi kiểm thử tuần tự nhưng tái hiện liên tục khi chạy stress test với 5 client đồng thời. GitHub Actions phát hiện lỗi này qua test case AuctionSessionConcurrencyTest ngay khi tôi push nhánh feature/bid-manager, trước khi mở PR.

Giải pháp: Sau khi phân tích qua thread #auction-engine trên Discord, tôi áp dụng synchronized block bao quanh phần kiểm tra và cập nhật highest bid trong BidManager, kết hợp ReentrantLock kiểm soát truy cập vào danh sách phiên đấu giá. Toàn bộ thay đổi được submit qua PR #5 với mô tả đầy đủ nguyên nhân và giải pháp, reviewer comment trực tiếp trên diff để xác nhận logic đúng trước khi merge. Quyết định thiết kế được ghi lại trong Decisions Log trên Google Docs để thành viên khác không vô tình tháo bỏ cơ chế khóa khi refactor.

Thách thức 2: Xung đột giao thức giữa Auction Engine và GUI

Thành viên phụ trách GUI bắt đầu code luồng hiển thị kết quả đấu giá dựa trên giả định về định dạng message từ server. Khi tích hợp, định dạng JSON response không khớp, mất gần nửa ngày để đồng bộ lại. Lỗi được phát hiện muộn vì hai nhánh feature/auction-engine và feature/gui-main-window phát triển song song mà không có điểm tích hợp sớm.

Giải pháp: Tôi soạn ngay Protocol Specification Document trên Google Docs với đầy đủ định dạng request/response cho từng sự kiện (PLACE_BID, AUCTION_END, NEW_HIGHEST_BID...), dùng Suggestion Mode để thành viên GUI review và phê duyệt trước khi implement. Đồng thời thiết lập quy trình: mọi thay đổi giao thức phải được cập nhật trên Docs và thông báo qua kênh Discord trước khi push code lên GitHub, ngăn tái diễn xung đột tương tự.

Thách thức 3: Phân tán quyết định kỹ thuật giữa Discord, GitHub và Google Docs

Nhiều quyết định quan trọng về cấu trúc database và cách lưu lịch sử giá thầu được thảo luận qua voice call Discord hoặc comment trên GitHub PR, nhưng không được tổng hợp lại. Thành viên phụ trách Data Layer implement theo cách khác với những gì đã thoả thuận, dẫn đến mâu thuẫn với logic trong BidManager và mất thời gian refactor.

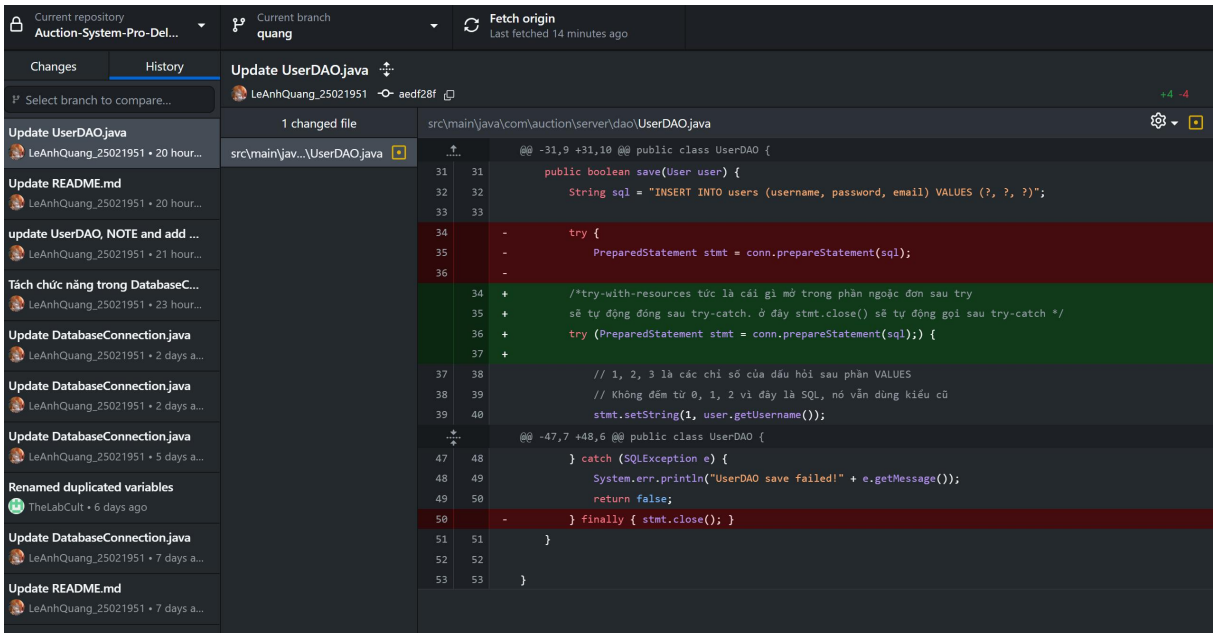
Giải pháp: Tôi thiết lập quy tắc cứng: mọi quyết định kỹ thuật phát sinh từ bất kỳ kênh nào (voice call, Discord, GitHub PR comment) đều phải được tổng hợp vào phần Decisions Log trong Google Docs trước khi cuộc thảo luận kết thúc, kèm tag @mention người chịu trách nhiệm. Webhook Discord tự động thông báo khi Docs có chỉnh sửa mới và khi GitHub có PR mở hoặc merged, giúp toàn nhóm nhận diện sớm thay đổi thiết kế mà không cần chủ động kiểm tra từng công cụ.

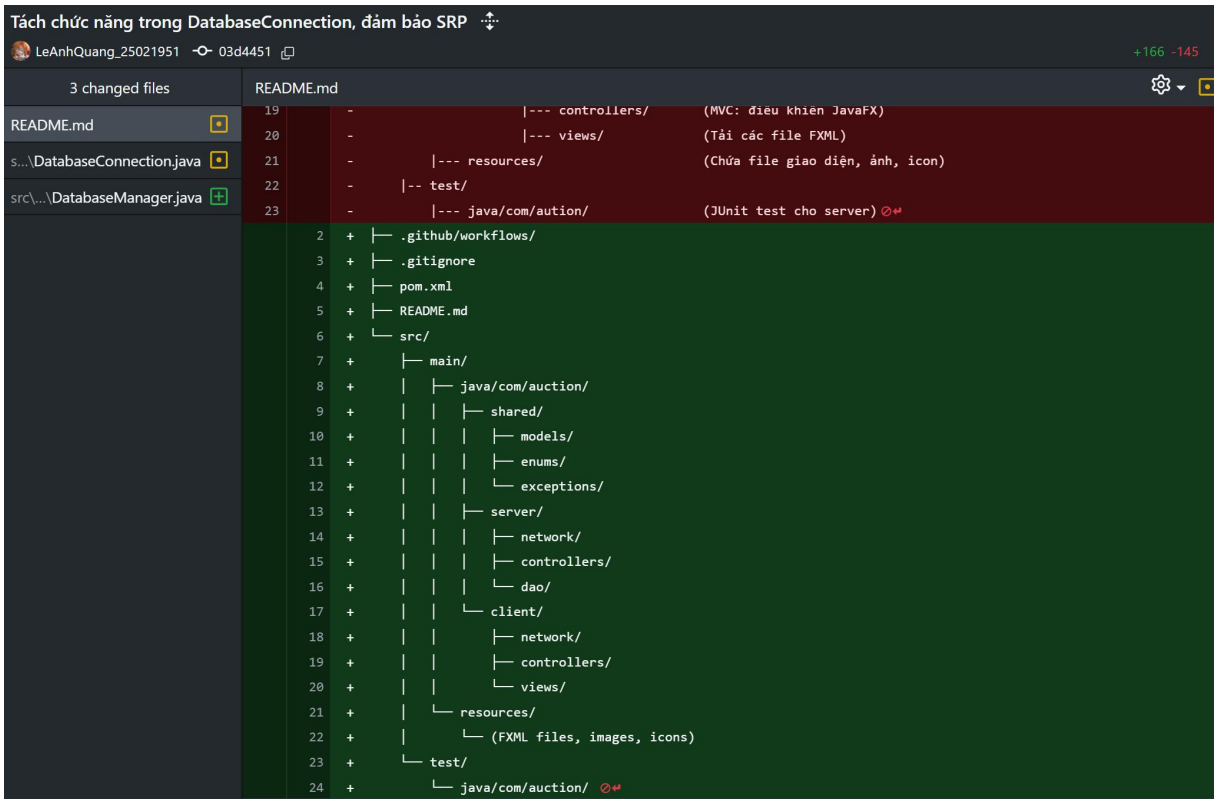
VI. KẾT LUẬN

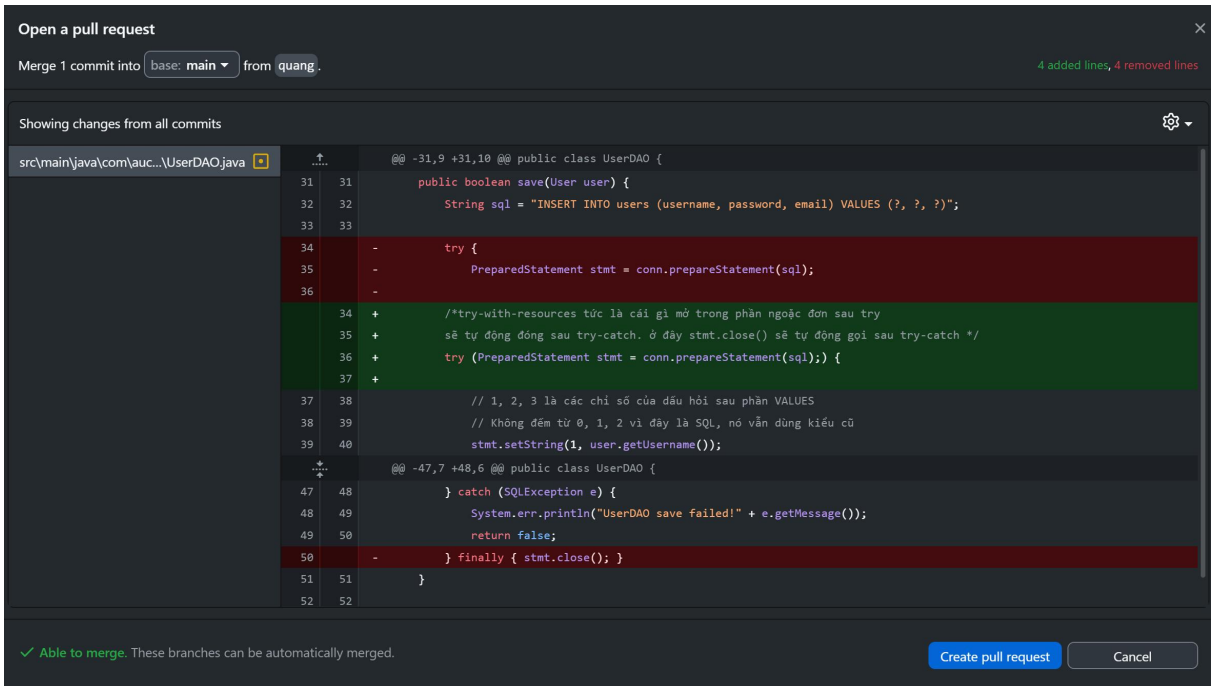
Qua 7 ngày triển khai dự án Auction App bằng Java, bộ 5 công cụ Trello – GitHub – Google Docs – Google Drive – Discord vận hành như một hệ sinh thái liền mạch, trong đó mỗi công cụ giải quyết đúng một nhóm nhu cầu: Trello phân công và theo dõi tiến độ; GitHub quản lý mã nguồn, review code và CI/CD tự động; Google Docs lưu giữ quyết định thiết kế; Google Drive tổ chức artifact; Discord kết nối thời gian thực. Việc tích hợp GitHub Actions phát hiện lỗi race condition ngay từ bước push, trước khi mở PR, là minh chứng rõ nhất cho giá trị của việc kết nối công cụ một cách có hệ thống.

Cá nhân tôi hoàn thành 100% nhiệm vụ đúng hạn, đóng vai trò chủ chốt trong thiết kế Auction Engine, xử lý 3 thách thức kỹ thuật và tổ chức quy trình làm việc cho cả nhóm. Kinh nghiệm này khẳng định rằng kỷ luật sử dụng công cụ và quy trình rõ ràng là yếu tố quyết định hiệu quả nhóm, quan trọng không kém năng lực kỹ thuật của từng cá nhân.

VII. DANH SÁCH MINH CHỨNG (ẢNH CHỤP MÀN HÌNH)




Tách chức năng trong DatabaseConnection, đảm bảo SRP
LeAnhQuang_25021951
03d4451
+166 -145
3 changed files
README.md
s...\DatabaseConnection.java
src\...\DatabaseManager.java
README.md
- |--- controllers/ (MVC: điều khiển JavaFX)
- |--- views/ (Tải các file FXML)
- |--- resources/ (Chứa file giao diện, ảnh, icon)
- |-- test/
- |--- java/com/aution/ (JUnit test cho server)
+ ├── .github/workflows/
+ ├── .gitignore
+ ├── pom.xml
+ ├── README.md
+ └── src/
+ ├── main/
+ │ ├── java/com/auction/
+ │ │ ├── shared/
+ │ │ │ ├── models/
+ │ │ │ ├── enums/
+ │ │ │ └── exceptions/
+ │ │ ├── server/
+ │ │ │ ├── network/
+ │ │ │ ├── controllers/
+ │ │ │ └── dao/
+ │ │ └── client/
+ │ │ ├── network/
+ │ │ ├── controllers/
+ │ │ └── views/
+ │ └── resources/
+ │ └── (FXML files, images, icons)
+ └── test/
+ └── java/com/auction/


Open a pull request
Merge 1 commit into base: main from quang.
4 added lines, 4 removed lines
Showing changes from all commits
src\main\java\com\auc...\UserDAO.java
@@ -31,9 +31,10 @@ public class UserDAO {
public boolean save(User user) {
String sql = "INSERT INTO users (username, password, email) VALUES (?, ?, ?)";
- try {
- PreparedStatement stmt = conn.prepareStatement(sql);
+ /*try-with-resources tức là cái gì mở trong phần ngoặc đơn sau try
+ sẽ tự động đóng sau try-catch. ở đây stmt.close() sẽ tự động gọi sau try-catch */
+ try (PreparedStatement stmt = conn.prepareStatement(sql);) {
// 1, 2, 3 là các chỉ số của dấu hỏi sau phần VALUES
// Không đếm từ 0, 1, 2 vì đây là SQL, nó vẫn dùng kiểu cũ
stmt.setString(1, user.getUsername());
@@ -47,7 +48,6 @@ public class UserDAO {
} catch (SQLException e) {
System.err.println("UserDAO save failed!" + e.getMessage());
return false;
- } finally { stmt.close(); }
}
Able to merge. These branches can be automatically merged.
Create pull request
Cancel

Tuần 8 – Kiểm thử, Ngoại lệ & GUI nâng cao

Bài giảng tuần này: Kiểm thử và Tái cấu trúc mã nguồn.
Công việc cụ thể:

1. Tạo custom exceptions: `InvalidBidException`, `AuctionClosedException`, `AuthenticationException`.
2. Xử lý ngoại lệ cho: đặt giá thấp hơn hiện tại, đấu giá khi phiên đóng, lỗi dữ liệu.
3. Viết unit test (JUnit) cho logic đấu giá: đặt giá hợp lệ/không hợp lệ, kết thúc phiên.
4. Refactor code: loại bỏ code smells, áp dụng SOLID.
5. **[TỰ HỌC]** Hoàn thiện GUI JavaFX – áp dụng MVC, tách logic khỏi Controller, dùng FXML.

Tài liệu học

- JUnit 5 User Guide: https://docs.junit.org/5.5.0/user-guide/

MVC trong JavaFX:

- Repo mẫu MVC: https://github.com/ZacharyDavidSaunders/InventoryManagementSystem
- Tài liệu MVC: https://www.pragmaticcoding.ca/javafx/MVC_In_JavaFX
- Video MVC: https://www.youtube.com/watch?v=Mu3VNzwpFII

Stellaaria dtth 07/04/2026 10:33 SA

Nội dung giảng dạy & Nội dung tự học

Phân biệt nội dung trên lớp và tự học

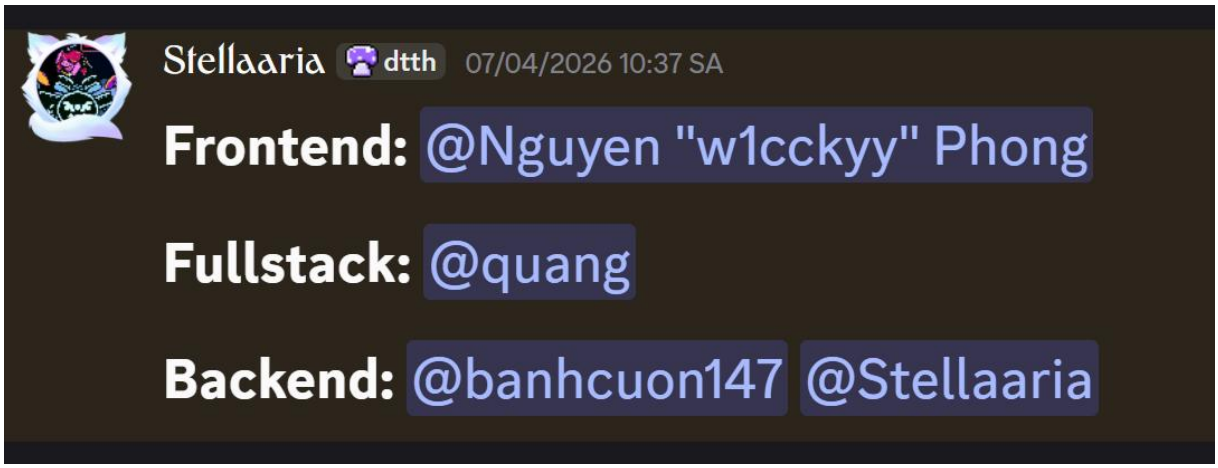
Khoá học có **10 bài giảng** (tuần 1–10). Từ tuần 11 trở đi **không còn bài giảng**.
Các nội dung sau **không được dạy trên lớp** mà sinh viên phải **tự học**:

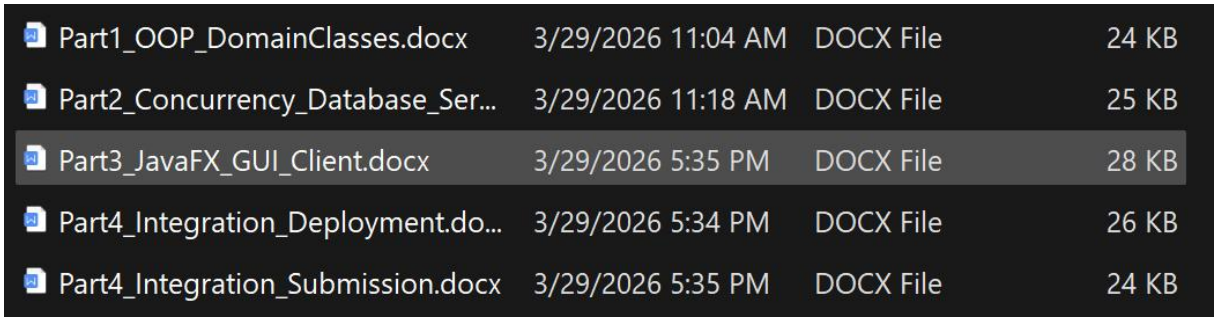
- **Lập trình mạng** (Socket, Client–Server)
- **Giao diện người dùng** (JavaFX, SceneBuilder, MVC)
- **Lưu trữ dữ liệu** (Serialization, File I/O)

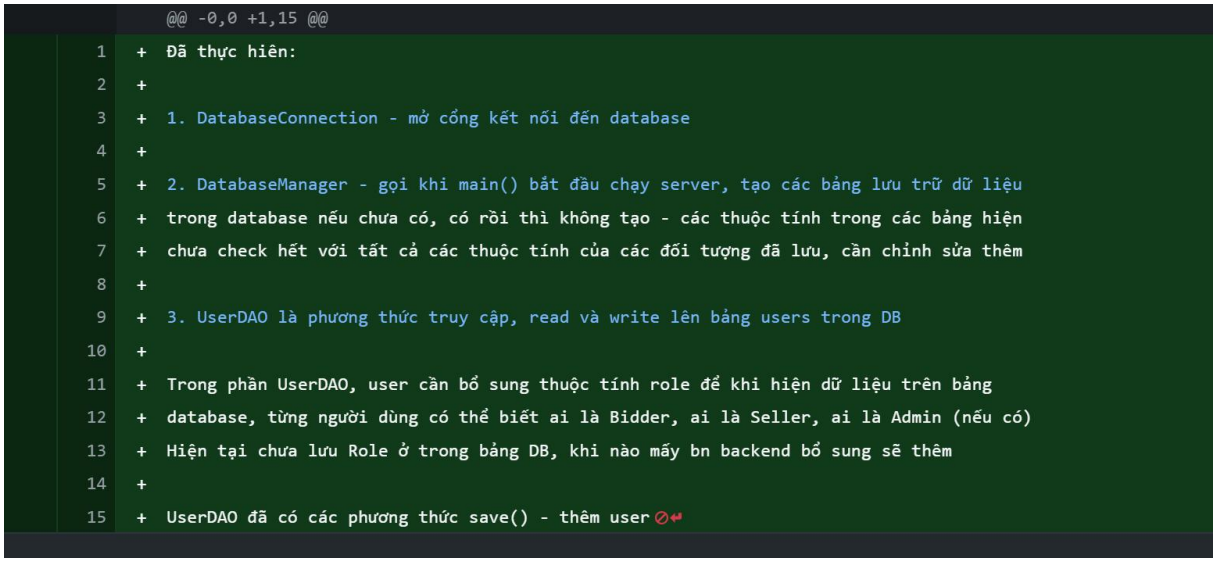
⇒ **Bắt đầu tự học các nội dung này ngay từ bây giờ**

Design Patterns: https://www.youtube.com/watch?v=mE3qTp1TEbg&list=PLlsmxlJgn1HJpa28yHzkBmUY-Ty71ZUGc
• Refactoring Guru: https://refactoring.guru/design-patterns
JavaFX – Học theo thứ tự sau:
1. Cài đặt JavaFX: https://openjfx.io/openjfx-docs/
2. Video setup IntelliJ: https://www.youtube.com/watch?v=Ope4icw6bVk
3. Playlist cơ bản (các component): https://www.youtube.com/watch?v=_7OM-cMYWbQ&list=PLZPZq0r_RZOM-8vJA3NQFZB7JroDcMwev
4. Setup SceneBuilder + IntelliJ: https://www.youtube.com/watch?v=IZCwawKILsk
5. Controller trong Scen v=0wQ68q7PD9w&list

Bạn Đang Xem Các Tin Nhắn Cũ | Nhảy đến hiện tại







```
@@ -0,0 +1,15 @@
+ Đã thực hiên:
+
+ 1. DatabaseConnection - mở cổng kết nối đến database
+
+ 2. DatabaseManager - gọi khi main() bắt đầu chạy server, tạo các bảng lưu trữ dữ liệu
+ trong database nếu chưa có, có rồi thì không tạo - các thuộc tính trong các bảng hiện
+ chưa check hết với tất cả các thuộc tính của các đối tượng đã lưu, cần chỉnh sửa thêm
+
+ 3. UserDAO là phương thức truy cập, read và write lên bảng users trong DB
+
+ Trong phần UserDAO, user cần bổ sung thuộc tính role để khi hiện dữ liệu trên bảng
+ database, từng người dùng có thể biết ai là Bidder, ai là Seller, ai là Admin (nếu có)
+ Hiện tại chưa lưu Role ở trong bảng DB, khi nào mấy bn backend bổ sung sẽ thêm
+
+ UserDAO đã có các phương thức save() - thêm user⊘↵
```