VIETNAM NATIONAL UNIVERSITY – HCM
INTERNATIONAL UNIVERSITY

SEMESTER 1 (2025-2026)

Lecturer: Nguyen Van Sinh

Lab Lecturer: Nguyen Trung Nghia

Web Application Development

TOPIC: Hyper bid web game

Team Members

Full Name	Student ID
Pham Trung Kien	ITCSIU23020

Github Repository:

https://github.com/Nobodywinsbutme/tyk-webapp

Project Report: Hyper bid web game

1 Abstract

This document is the project report template for the Web Application Development course. It describes the system purpose, requirements, architecture, API list, database design (ERD), user/workflow diagrams, testing plan, deployment guide, and appendices. The report includes placeholders and blank areas where you can insert ERD images, workflow diagrams, screenshots, and additional notes.

2 Table of contents

3 Project overview

3.1 Project name

Hyper bid web game

3.2 Short description

A skeleton hyper-website styled bidding game where users can register/login, view items, place bids, see leaderboards and play interactive micro-games. Backend implemented in Java; frontend uses HTML/JavaScript.

4 Requirements

4.1 Functional requirements

- User registration and authentication (JWT).
- Browse available items/games.
- Place bids and view bid history.
- Real-time updates for live bidding (WebSocket or polling).
- View leaderboard and user profiles.
- Admin panel for item management.

4.2 Non-functional requirements

- Responsive UI (desktop/mobile).
- Secure authentication and authorization.
- Scalable backend to handle concurrent bids.
- Persistent storage (database) for users, items, bids.
- CI/CD pipeline for automated builds and deployments.

5 System architecture

Include a high-level architecture diagram here (use draw.io, or insert an image). Example components:

- Client (Browser) - HTML/JS
- Backend API - Java (Spring Boot / alternative)
- Database - PostgreSQL / MySQL
- Real-time layer - WebSocket / Socket.IO
- Reverse proxy - Nginx (TLS)
- CI/CD - GitHub Actions / Docker registry

Architecture diagram:

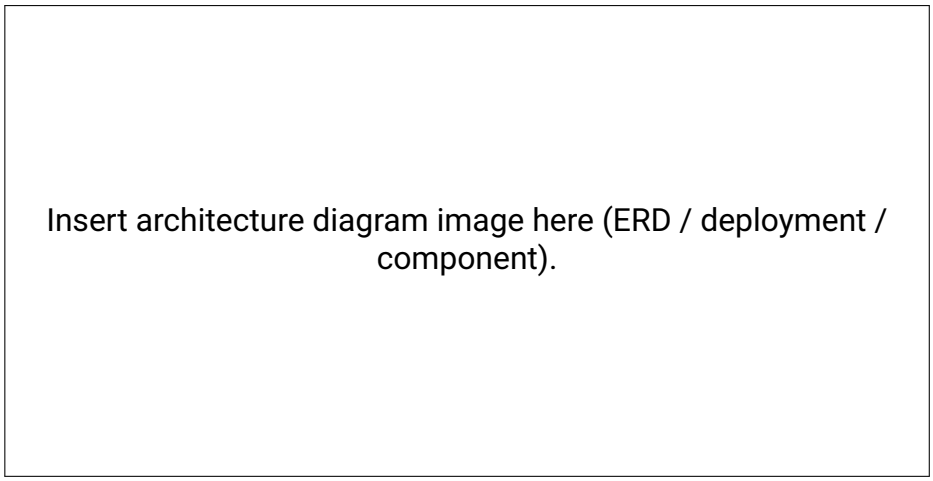

Figure 1: High-level architecture (placeholder).

6 Technology stack

Backend	Java (recommended: Spring Boot)
Frontend	HTML, JavaScript (vanilla or framework like React/Vue)
Database	PostgreSQL / MySQL
Real-time	WebSocket or Server-Sent Events
Build tools	Maven / Gradle, npm/yarn
Containerization	Docker, Docker Compose
CI/CD	GitHub Actions (recommended)

7 API list

Below is a structured API summary. Fill or extend entries as needed.

Endpoint	Method	Description	Auth
/api/auth/register	POST	Create new user account	No
/api/auth/login	POST	Authenticate user, return JWT	No
/api/users/me	GET	Get current user profile	Yes (Bearer)
/api/items	GET	List items available for bidding	No
/api/items	POST	Create new item (admin)	Yes (Admin)
/api/items/id	GET	Get item detail	No
/api/bids	POST	Place a bid on an item	Yes
/api/bids/item/itemId	GET	Get bid history for item	No
/api/games/start	POST	Start a mini-game session	Yes
/api/leaderboard	GET	Get leaderboard data	No
/api/admin/stats	GET	Admin stats	Yes (Admin)

Table 1: Summary of core APIs

7.1 Example: Register

```
POST /api/auth/register
Content-Type: application/json

{
  "username": "alice",
  "email": "alice@example.com",
  "password": "P@ssw0rd"
}
```

Listing 1: Request example - Register

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "id": 123,
  "username": "alice",
  "email": "alice@example.com",
  "createdAt": "2025-12-21T10:00:00Z"
}
```

Listing 2: Response example - Register

7.2 Example: Login

```
POST /api/auth/login
Content-Type: application/json

{
  "username": "alice",
  "password": "P@ssw0rd"
}
```

Listing 3: Request example - Login

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "expiresIn": 3600
}
```

Listing 4: Response example - Login

7.3 API details table (expand in appendix)

Use the appendix to add detailed parameter, response schema, error codes, and examples.

8 Database design (ERD)

Provide ERD here. Below is a simple suggested schema; draw actual ERD and paste an image.

8.1 Suggested entities

- User (id, username, email, passwordHash, role, createdAt)
- Item (id, title, description, startPrice, currentPrice, startAt, endAt, ownerId)
- Bid (id, itemId, userId, amount, createdAt)
- GameSession (id, userId, gameType, score, createdAt)
- Audit / Logs

ERD placeholder:

Figure 2: Entity Relationship Diagram (placeholder).

Draw/Insert ERD here

9 Workflows and user journeys

Include sequence diagrams or flowcharts describing the main flows:

- User registration and login
- Browsing items and placing a bid
- Real-time auction flow
- Admin item lifecycle (create/edit/remove)

Workflow placeholder:

Insert workflow / sequence diagram here.

Figure 3: Main user workflow (placeholder).

10 UI / Screenshots

Add screenshots of important pages (login, item list, item detail, bidding modal, leaderboard, admin area). Insert images with `\includegraphics[]{}`.

Example placeholder:

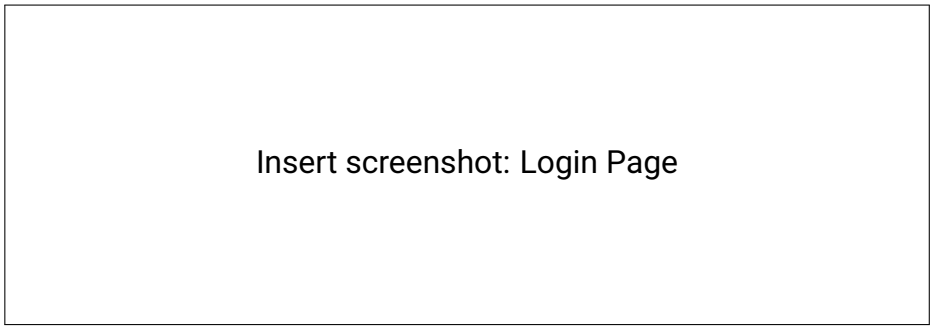

Figure 4: Login page (placeholder)

11 Deployment and operations

11.1 Local development

1. Clone repository: `git clone https://github.com/Nobodywinsbutme/tyk-webapp.git`
2. Backend: build with Maven/Gradle and run.
3. Frontend: open index.html or run dev server (npm).
4. DB: use Docker or local Postgres.

11.2 Production

- Containerize backend and frontend using Docker.
- Use managed DB (RDS / Cloud SQL).
- Setup CI to build and run tests, then push images to registry.
- Use a reverse proxy (nginx) to terminate TLS.

12 Testing plan

- Unit tests: Java (JUnit), JS (Jest).
- Integration tests: run against a test DB (Docker Compose).
- E2E: Cypress / Playwright for critical flows (login, bidding).
- Load test: simulate concurrent bids (k6, JMeter).

13 Security considerations

- Store passwords as salted hashes (bcrypt / Argon2).
- Use HTTPS/TLS in production.
- Protect against race conditions in bidding (DB transactions / optimistic locking).
- Validate and sanitize all inputs.
- Rate-limit public endpoints and login attempts.
- Use JWT with short expiry and refresh token pattern.

14 Project plan and timeline

Provide a Gantt or simple timeline here. Example milestones:

- Week 1: Requirements design
- Week 2-3: Backend core APIs and DB models
- Week 4: Frontend UI and integration
- Week 5: Real-time features and testing
- Week 6: Polish, deployment, final report

15 Contributions

- Pham Trung Kien – Backend, API, database design

16 References

- Project repo: https://github.com/Nobodywinsbutme/tyk-webapp
- Course materials and slides
- Library docs: Spring Boot, PostgreSQL, Docker

Appendix A: Detailed API documentation

Use this section to expand each endpoint with parameters, responses, status codes, sample cURL and error examples.

Example: Place Bid (detailed)

Endpoint: POST /api/bids
Auth: Bearer token
Request:

```
POST /api/bids
Authorization: Bearer <token>
Content-Type: application/json

{
  "itemId": 42,
  "amount": 150.00
}
```

Response (success):

```
HTTP/1.1 201 Created
{
  "bidId": 987,
  "itemId": 42,
  "userId": 123,
  "amount": 150.00,
  "createdAt": "2025-12-21T10:05:00Z"
}
```

Errors:

- 400 Bad Request - invalid amount
- 401 Unauthorized - missing/invalid token
- 409 Conflict - bid lower than current highest or auction closed

Appendix B: Sample cURL commands

```
# Login and get token
curl -X POST https://your-host/api/auth/login -H "Content-Type:
   application/json" \
```

```
  -d '{"username":"alice","password":"P@ssw0rd"}'

# Place a bid
curl -X POST https://your-host/api/bids \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/json" \
  -d '{"itemId":42,"amount":150.00}'
```