Team roles and responsibilities

Person	Role
Thanh, Project Manager	Responsible for the operation and maintenance of the project. Managing project plans and schedules. Also responsible for the deployment and architecture.
Tú, Senior System Engineer	Responsible for the system integration and designing, including backend and frontend
Nam, Senior Software Engineer	Responsible for the development and building of the frontend and backend system of the product.
Nhân, Software Engineer	Responsible for the development and building of the frontend and backend system of the website.
Nghĩa, Tester	Responsible for the testing system, reporting system errors and issues before deploying the product to end-users.

Team communications

Google Drive

- Used to store and share all documentation (e.g., sprint plans, reports, design documents).
- Organized by sprint phases to ensure easy access and version control.

GitHub

- Hosts the source code and manages version control through branches and pull requests.
- Tasks and progress will be tracked using GitHub Issues, with all code requiring peer review before merging.

Slack

- Serves as the primary communication platform for discussions, announcements, and coordination.
- Channels will be created for different topics to keep conversations organized (e.g., `#frontend`, `#testing`).

Jira:

- Used for task management and tracking sprint progress.
- Tasks will be assigned to team members, with clear deadlines and dependencies.
- Provides real-time tracking of progress using Kanban boards and sprint reports.

Work schedule and deadlines

Project milestones and deadlines

Milestone	Sprint	Tasks	Start Date	End Date
Research phase	Sprint 1	Research on frontend (React) and backend (Node.js)	21 Oct 2024	2 Nov 2024
Design & planning	Sprint 2	Create use-case diagrams, UI design, and update vision document	3 Nov 2024	16 Nov 2024
Development phase 1	Sprint 3	Code UI, database, and design architecture documents	17 Nov 2024	30 Nov 2024
Development phase 2	Sprint 4	Code main functions, fix bugs, and update documentation	1 Dec 2024	14 Dec 2024
Testing and deployment	Sprint 5	Perform testing, finalize functions, and release product	15 Dec 2024	28 Dec 2024

Team availability

- **Meeting schedule**:
  - Weekly **status meetings** every **Monday at 10 AM**.
  - Sprint **planning meetings** are held on the first Monday of each sprint.
  - **Ad-hoc work sessions** scheduled based on member availability.
- **Team member availability**:
  - **Thanh** (Project Manager): Works across multiple time slots daily for coordination, sprint planning, and stakeholder updates, often extending hours during critical phases.
  - **Tú** (Senior System Engineer): Engaged from mid-morning to late evening on system integration, architectural reviews, and evening syncs. Available for weekend tasks if needed.
  - **Nam** (Senior Software Engineer): Focuses on afternoons, evenings, and weekends for development, debugging, and sprint preparation, often working late into the night.

    - **Nhân** (Software Engineer): Splits time across mornings, afternoons, and late nights, assisting development and attending syncs with Nam, with weekend availability when required.
    - **Nghĩa** (Tester): Fully occupied with testing during afternoons and late nights, coordinating evening debugging and weekend test cycles for smooth release.

Contingency plans for missed deadlines or delays

- **Reporting delays**:
    - Any potential delays must be reported to Thanh (Project Manager) **immediately** to allow adjustments in the schedule.
- **Task reassignment**:
    - If a member cannot complete a task on time, tasks will be **reassigned** to other available members.
    - Thanh will adjust the priority list and **allocate additional time** during weekends for recovery work if needed.
- **Buffer time**:
    - Each sprint includes **2-3 days buffer** to absorb any unexpected delays or additional work.
- **Risk mitigation**:
    - If tasks require external dependencies (e.g., tools, third-party APIs), the team will initiate **early testing and backups**.
    - Conflicts between team members will be resolved swiftly during meetings, with the final decision resting with the Project Manager.

Code and Documentation Standards

Coding conventions and tools

- **Frontend**:
  - Use **React** for frontend development.
  - Follow **Airbnb's JavaScript Style Guide** for consistent JavaScript/React code.
  - Use **ESLint** for static code analysis to detect errors and enforce coding standards.
  - Naming conventions:
    - Variables: camelCase (e.g., `productList`)
    - Components: PascalCase (e.g., `ProductCard.js`)
    - Constants: UPPER_CASE (e.g., `MAX_DISCOUNT`)
- **Backend**:
  - Use **Node.js** and **Express** for server-side development.
  - Use **Prettier** for code formatting to maintain readability and consistency.
  - Follow RESTful API design principles for backend endpoints.
  - Implement **JWT Authentication** for secure user sessions.
- **Database**:
  - Use **MongoDB** for database management.
  - Follow clear collection naming conventions in plural (e.g., `users`, `orders`).
  - Use **Mongoose** ORM for database modeling.
- **Version Control**:
  - Use **Git** for version control. The main branch will always be stable and production-ready.
  - **Branching Strategy**:
    - Use feature branches (e.g., `feature/login-page`) for new features.
    - Use hotfix branches (e.g., `hotfix/order-bug`) for urgent bug fixes.

Code review and testing procedures

- **Code reviews**
  - Every pull request (PR) must be reviewed by at least **one peer** before being merged into the main branch.
  - Reviewers should check for:
    - Code readability and maintainability.

        - Consistency with coding standards.
        - Absence of unnecessary or unused code.
        - Proper handling of edge cases.
- **Testing procedures**
    - **Unit tests**: Implement unit tests using **Jest** for both frontend and backend logic.
    - **Integration tests**: Use **Postman** to test API endpoints and backend interactions.
    - **UI testing**: Use **Cypress** for automated testing of frontend components.
    - **Test reporting**: Nghĩa (Tester) will maintain a test log with the following:
        - Defects discovered during each sprint.
        - Status of tests (open/closed).
        - Estimated time to resolve reported issues.

Project documentation guidelines

- **Documentation tools**:
    - Use **Markdown** files (`README.md`) in each module directory to explain its functionality.
    - Use **Jira** (or similar tools) to track issues, progress, and feature development.
- **Documentation requirements**:
    - **Code documentation**:
        - Comment all public methods with descriptions of their purpose and parameters.
        - Use JSDoc-style comments for JavaScript functions.
        - Add in-line comments where necessary to clarify complex logic.
    - **Project documentation**:
        - **Architecture document**: Describe the system's architecture and technology stack.
        - **API documentation**: Provide a Postman collection or OpenAPI (Swagger) specification for all backend APIs.
        - **User documentation**: Include a **User Manual** explaining how to use the website, covering both customer and retailer operations.
- **Versioning and updates**:
    - Documentation must be updated with every new release to reflect any changes in the system.
    - Each document will have a **version history** table tracking modifications, with the author and date noted.

Accountability and Performance

Criteria for measuring contribution and quality of work

- **Task completion**:
  - Tasks assigned in sprints must be completed within the given deadline, with progress reported weekly.
- **Code quality**:
  - Adherence to coding standards (using ESLint, Prettier) and passing all code reviews.
- **Testing success rate**:
  - All implemented features must pass unit and integration tests, with defect rates tracked.
- **Participation in meetings and syncs**:
  - Attendance and active participation in team meetings, including sprint planning and daily stand-ups.
- **Documentation**:
  - All work must be properly documented, including code comments and updates to project documentation (e.g., architecture or user manuals).

Process for handling underperformance or lack of participation

- **Step 1: Verbal warning**
  - The Project Manager (Thanh) will notify the member about the performance issue in a private discussion and provide support if needed.
- **Step 2: Formal review**
  - If the issue persists, a formal review will be conducted during the next sprint planning session. The member will need to provide a reason for underperformance and propose an improvement plan.
- **Step 3: Task redistribution**
  - If no improvement is seen after the formal review, tasks will be redistributed among other members to avoid delays.

Consequences for not adhering to the team contract

- **Reduced role or responsibilities**:
  - Persistent underperformance or absence may result in the reassignment of tasks and exclusion from key project phases.
- **Reduced credit or recognition**:
  - Members who fail to meet participation standards will receive reduced recognition or credit in the project’s final submission.

- **Escalation to academic authorities** (if applicable):
    - In case of major conflicts or severe lack of contribution, the issue will be escalated to the project supervisor or course instructor.

Decision - Making Process

Decision-Making approach

- **Consensus as the primary method**:
    - The team will aim to reach a consensus on all decisions during meetings, ensuring that every member's perspective is heard and considered.
    - Discussions will focus on the project's objectives, feasibility, and alignment with timelines.
- **Use of majority vote**:
    - If consensus cannot be achieved within a reasonable timeframe, the decision will proceed to a **majority vote**.
    - Each team member will cast a vote, and the option with the most votes will be implemented.

Final decision authority

- **Project manager's final say**:
    - In the event of unresolved disagreements after voting or discussions, **Thanh (Project Manager)** will make the final decision to avoid project delays.
    - Thanh's decisions will prioritize the project's success, alignment with deadlines, and resource management.

Conflict Resolution

To maintain a healthy working environment and ensure project success, a framework for resolving conflicts between team members will be established:

- Team members are encouraged to discuss disagreements or misunderstandings directly and promptly. An open dialogue should be the first step toward resolving any issues.
- If a conflict cannot be resolved through open communication, the team lead will mediate a discussion to understand the root of the issue and suggest possible solutions.
- In cases where mediation does not resolve the conflict or the issue is significant, the matter will be escalated to the teacher for further resolution. The supervisor will work with the involved parties to reach a fair solution, which may include adjustments in roles or responsibilities if necessary.
- All resolutions will be documented to prevent similar conflicts in the future and provide a reference for future conflict resolution processes.

Review and Update Process

To ensure the contract remains relevant and aligned with the project's evolving needs, a systematic review and update process will be implemented:

- The contract will be reviewed at predefined intervals, such as every three months or at the completion of key project phases, to assess its effectiveness and alignment with project goals.
- Feedback from team members will be actively sought during reviews. Any proposed changes will be discussed collectively and incorporated with mutual agreement.
- The contract will allow for adjustments at any time if unforeseen circumstances arise, ensuring the team can respond promptly to changing project dynamics.
- All changes to the contract must be approved by the team lead and supervisor to ensure that updates are in the project's best interest and do not conflict with existing agreements or regulations.