

**CAPSTONE PROJECT REPORT**

**Report 2 – Project Management Plan**

– Hanoi, April 2025 –

**Table of Contents**

[I. Record of Changes 3](#_heading=h.21qt7m95yqgu)

[II. Project Management Plan 4](#_heading=h.eu660eida8m4)

[1. Overview 4](#_heading=h.lyviksr2kp7d)

[1.1 Cost & Time Estimations 4](#_heading=h.eqme2wee7ime)

[1.2 Project Objectives 5](#_heading=h.j3qot7p24hp9)

[1.3 Project Risks 6](#_heading=h.awjhskslrnoj)

[2. Management Approach 6](#_heading=h.46kfhz4l1hmn)

[2.1 Project Process 7](#_heading=h.xw1ghme52lgq)

[2.2 Quality Management 8](#_heading=h.cyj80k5v9sk8)

[2.2 Project Training Plan 10](#_heading=h.t3z99i5a7z3m)

[3. Responsibility Assignments 10](#_heading=h.jfxelr9inpzm)

[4. Project Communications 13](#_heading=h.6ug5zgusljuv)

[5. Configuration Management 14](#_heading=h.lnsm13u85fc9)

[5.1 Document Management 14](#_heading=h.xct7llhwyik)

[5.2 Source Code Management 14](#_heading=h.aqjjzqql5l1d)

[5.3 Tools & Infrastructures 15](#_heading=h.6wzjxbq7828)

# I. Record of Changes

| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

# II. Project Management Plan

## 1. Overview

### 1.1 Cost & Time Estimations

| **#** | **Work Package** | **Est. Effort**  **(pds)** | **Deadline** |
| --- | --- | --- | --- |
| ***1*** | ***Stage 1: Project Initiating (week 1)*** | ***10*** | ***12-01-2025*** |
| 1.1 | Report 1 (Project Introduction) | 10 | 12-01-2025 |
| ***2.*** | ***Stage 2: Project Planning & Initial Requirements (weeks 2-3)*** | ***50*** | ***06-02-2025*** |
| 2.1 | Report 3: SRS v0.9 (Overall Requirement) | 15 | 06-02-2025 |
| 2.2 | Vertical Prototypes (Code Demo) | 15 | 30-01-2025 |
| 2.3 | Report 2: Project Management Plan v1.0 | 20 | 25-01-2025 |
| ***3*** | ***Stage 3: Software Design (weeks 4-5)*** | ***60*** | ***20-02-2025*** |
| 3.1 | Vertical Prototypes (POC, or Proof Of Concepts) | 20 | 20-02-2025 |
| 3.2 | Report 4: SDS v1.0 (High Level Design Specifications) | 15 | 20-02-2025 |
| 3.3 | Report 5: Test Planning (ref Report5\_Test Documentation) | 10 | 20-02-2025 |
| 3.4 | Report 3: SRS v1.0 (Detailed Requirements for Iteration 1) | 15 | 20-02-2025 |
| ***4*** | ***Stage 4: Implementation (weeks 6-11)***  (Ref: [Project Tracking](about:blank), WBS sheet) | ***175*** | ***03-04-2025*** |
| 4.1 | Implementation Iteration 1 (weeks 6-7) | 55 | 04-03-2025 |
| 4.1.1 | Report 4: Detailed Design Specifications (SDS) v1.1 | 10 | 21-02-2025 |
| 4.1.2 | Code Package & Unit Testing Report | 15 | 04-03-2025 |
| 4. 1.3 | Report 5: Integration Test Cases | 10 | 04-03-2025 |
| 4. 1.4 | Report 5: Integration Test Report | 5 | 04-03-2025 |
| 4.1.5 | Report 3: SRS v1.1 (Detailed Requirements for Iteration 2) | 10 | 04-03-2025 |
| 4.1.6 | Software Package Version 1 | 5 | 04-03-2025 |
| 4.2 | Implementation Iteration 2 (weeks 8-9) | 55 | 18-03-2025 |
| 4.2.1 | Report 4: Detailed Design Specifications (SDS) v1.2 | 10 | 10-03-2025 |
| 4.2.2 | Code Package & Unit Testing Report | 15 | 18-03-2025 |
| 4. 2.3 | Report 5: Integration Test Cases | 10 | 18-03-2025 |
| 4. 2.4 | Report 5: Integration Test Report | 5 | 18-03-2025 |
| 4.2.5 | Report 3: SRS v1.2 (Detailed Requirements for Iteration 3) | 10 | 18-03-2025 |
| 4.2.6 | Software Package Version 2 | 5 | 18-03-2025 |
| 4.3 | Implementation Iteration 3 (weeks 10-11) | 55 | 03-04-2025 |
| 4.3.1 | Report 4: Detailed Design Specifications (SDS) v1.3 | 10 | 21-03-2025 |
| 4.3.2 | Code Package & Unit Testing Report | 15 | 03-04-2025 |
| 4.3.3 | Report 5: Integration Test Cases | 10 | 03-04-2025 |
| 4.3.4 | Report 5: Integration Test Report | 5 | 03-04-2025 |
| 4.3.5 | Report 5: System Test Cases | 10 | 03-04-2025 |
| 4.3.6 | Software Package Version 3 | 5 | 03-04-2025 |
| ***5*** | ***Stage 5: Verification & Validation (weeks 12-13)*** | ***55*** | ***17-04-2025*** |
| 5.1 | Report 5: System Test Report | 15 | 10-04-2025 |
| 5.2 | Report 5: Acceptance Test Support | 10 | 10-04-2025 |
| 5.3 | Report 6: User Guides (Installation Guides, User Manuals) | 15 | 17-04-2025 |
| 5.4 | Final Software Package (Documents, Source Codes, etc.) | 15 | 17-04-2025 |
| ***6*** | ***Stage 6: Closing (week 14)*** | ***25*** | ***24-04-2025*** |
| 6.1 | Report 7: Final Project Report | 15 | 20-04-2025 |
| 6.2 | Presentation File | 10 | 22-04-2025 |
| 6.3 | Practice for presentation of Capstone Project | 10 | 24-04-2025 |
| Total | | 375 |  |

### 1.2 Project Objectives

### 

| **#** | **Metric** | **Unit** | **Planned** | **Actual** | **Notes / References** |
| --- | --- | --- | --- | --- | --- |
| 1 | Effort Usage | Person-day | 280 |  |  |
| 2 | Review Defects | No of defects | 100% |  |  |
| 3 | Unit Test Defects | No of defects | 100% |  |  |
| 4 | Integration Test Defects | No of defects | 100% |  |  |
| 5 | System Test Defects | No of defects | 100% |  |  |
| 6 | Acceptance Test Defects | No of defects | 100% |  |  |
| 7 | Timeliness | % | 95% |  | Total on-time deliverables / Total project deliverables |
| 8 | Requirement Completeness | % | 100% |  | Failed if <=75% |

### 1.3 Project Risks

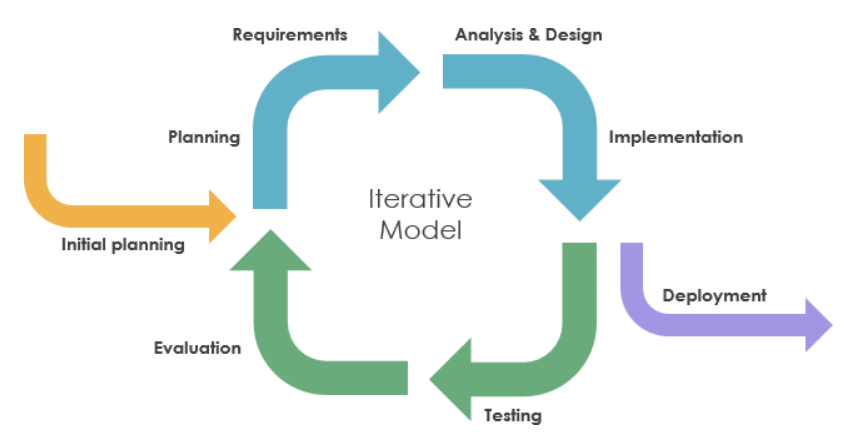
| **#** | **Risk Description** | **Impact** | **Possibility** | **Response Plans** |
| --- | --- | --- | --- | --- |
| 1 | Members have argued, conflicted with others, leads to stressful working environments. | Medium | High | Define clear tasks for each member and agree on ideas before starting work. |
| 2 | Geographical Restrictions | Medium | High | Expand the service area gradually. Conduct market research to identify potential new areas for expansion. |
| 3 | Payment Gateway Issues | High | Low | Partner with reliable payment gateway providers. Have backup payment options available. |
| 4 | Limited Customer Support | Medium | High | Members engage in discussions regarding the prioritisation of requirements desired by the staff for modification and addition. Subsequently, these requirements are incorporated into each sprint to execute high-priority tasks. |

## 

## 2. Management Approach

The Pickleball Court Management System development management methodology is built on a structured process to ensure that the project achieves its objectives effectively. This process includes specific steps, detailed in the Project Process section below:

### 2.1 Project Process

  
  
The Pickleball Court Management System project development process includes the following steps:

#### 2.1.1. Initiation and Planning

* **Define Core Objectives**: The system focuses on automated court management, enabling owners of multiple courts and small businesses to optimize operations, monitor court status, manage memberships, and automate payments.
* **Identify Key Features**: Ensure an easy-to-use interface for court owners, seamless integration for tracking court status (in use, empty, maintenance), and generating revenue reports. Court booking is a secondary feature.
* **Gather Insights**: Collect feedback from court owners and staff to understand needs, such as automating schedules and reducing manual errors.

#### 2.1.2. Requirements Analysis

* **Collect and Document Needs**: Survey court owners to identify issues like difficulty tracking court status, manual payment management, and revenue reporting.
* **Prioritize Features**: Focus on features like real-time court status tracking, payment automation, and reporting, with court booking as a supporting function.
* **Develop Requirements Documentation**: Detail functional requirements (e.g., managing court maintenance schedules) and non-functional requirements (e.g., user-friendly interface).

#### 2.1.3. System Design

* **Plan in Detail**: Design the system architecture centered on court management, with a database for storing court, payment, and reporting data.
* **Build Models**: Create a management dashboard for court owners (e.g., displaying court status) and a database schema for tracking court usage history.
* **Ensure Flexibility**: Design the system to be scalable for managing additional courts or integrating new features in the future.

#### 2.1.4. Development

* **Build the System**: Use web technologies to develop components like the court management dashboard, reporting system, and payment automation.
* **Iterative Approach**: Develop in phases, with each phase focusing on a specific feature, such as real-time court status display or revenue reporting.
* **Incremental Testing**: Ensure each management feature works well before integrating with the secondary booking function.

#### 2.1.5. Testing

* **Testing Strategy**: Conduct unit testing for individual management features, integration testing to ensure components work together, and user acceptance testing with court owners.
* **Gather Feedback**: Request input from court owners to improve, such as speeding up court status updates or simplifying the reporting interface.
* **Ensure Quality**: Thoroughly test to ensure reliability, minimizing errors in management tasks.

#### 2.1.6. Deployment

* **Launch the System**: Deploy the application after completing core court management features, meeting the needs of court owners and small businesses.
* **Support Users**: Provide guides and training for court owners to use the system effectively.
* **Monitor Operations**: Set up tools to monitor performance and detect issues, such as errors during peak times.

#### 2.1.7. Maintenance and Support

* **Maintain Operations**: Regularly monitor and update the system to ensure performance and security.
* **Support Users**: Offer communication channels to address queries from court owners and staff.
* **Plan for Upgrades**: Anticipate enhancements, such as adding court usage analytics or support for more courts.

### 2.2 Quality Management

#### 2.2.1 Defect Prevention

* Clearly communicate project tasks and challenges, such as automating court status tracking and payment workflows, to all team members during initial meetings.
* Ensure all team members fully understand the requirements, like real-time court monitoring and revenue reporting, by reviewing them together in detail.
* Assign roles based on strengths: developers with database expertise handle court data management, while UI specialists design the owner dashboard.
* Maintain consistent roles throughout the project to streamline collaboration and reduce confusion.

#### 2.2.2 Reviewing

* Regularly review all project components, including requirements documents, code for court management features, and development schedules, to ensure alignment with goals.
* Identify and list any issues, such as errors in payment automation or discrepancies in court status updates, and prioritize them based on impact (e.g., critical bugs affecting revenue reports are top priority).
* Conduct frequent peer reviews to catch potential issues early, ensuring high-quality deliverables.

#### 2.2.3 Unit Testing

* Select appropriate testing frameworks, such as JUnit for backend court management logic and Test for frontend components like the owner dashboard.
* Create multiple test cases for different modules, including court status tracking, payment processing, and revenue reporting.
* Review test cases for errors, fixing issues like incorrect assertions, and ensure they cover all scenarios, such as handling peak booking times.
* Verify that every line of code adheres to the project’s Standard Pattern Development guidelines for consistency and quality.
* Assign the developer responsible for any defects to resolve them quickly, ensuring minimal delays in the development timeline.

#### 

#### 2.2.4 Integration Testing

* Begin integration testing only after all individual components, such as court scheduling and payment automation, have passed unit testing.
* Test components in a logical sequence: first, ensure court status updates work with the scheduling system, then integrate payment processing to confirm seamless operation.
* Document any issues, such as payment failures affecting court bookings, and assign them to the relevant team members for resolution, retesting after fixes to confirm functionality.

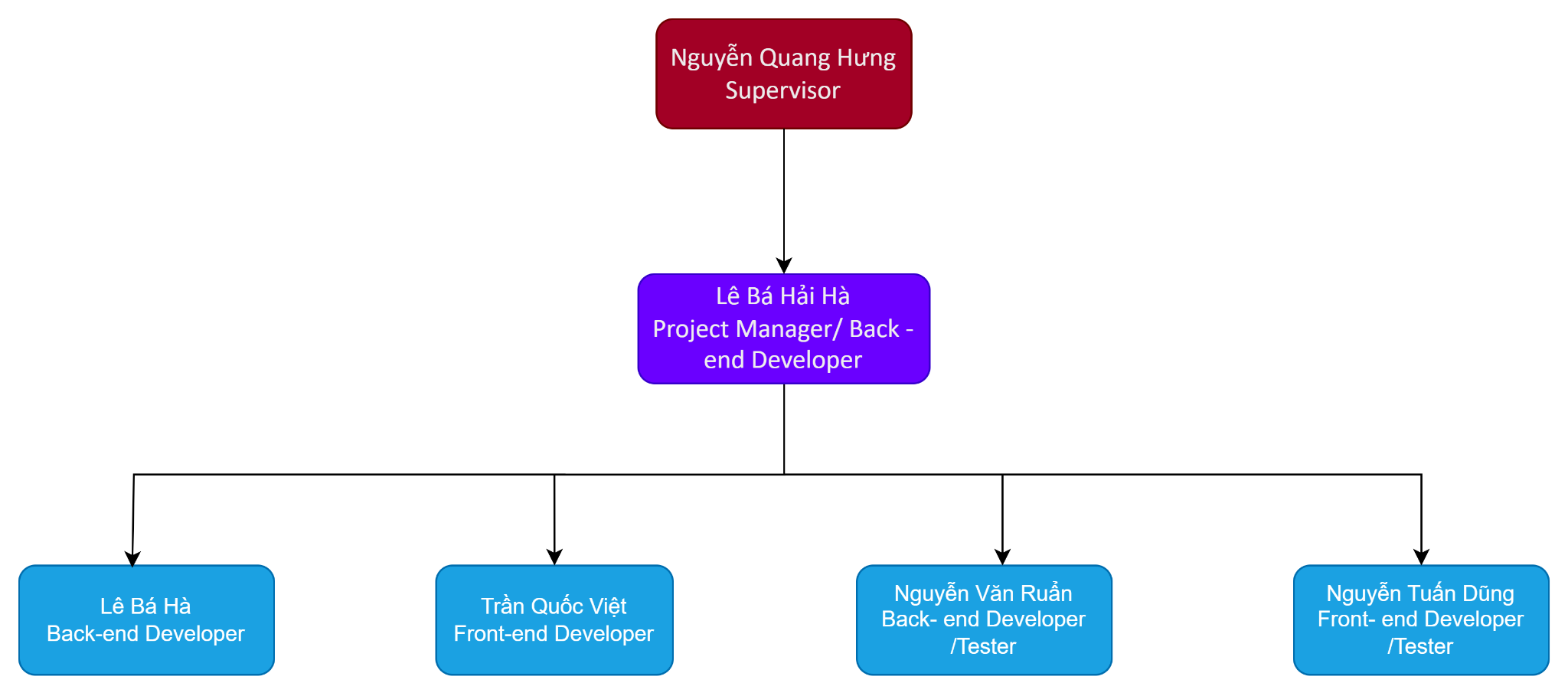
#### 2.2.5 System Testing

* Develop a testing plan with clear start and end criteria, such as completing all integration tests before starting and passing all test cases before finishing.
* Validate the entire system to ensure it meets requirements, like accurately tracking court usage and generating revenue reports for owners.
* Identify defects, such as delays in court status updates or errors in financial reports, and address them promptly.
* Confirm the system is ready for deployment, ensuring it supports court owners’ operational needs with minimal downtime and a user-friendly experience.

### 2.2 Project Training Plan

| **Training Area** | **Participants** | **When, Duration** |
| --- | --- | --- |
| ReactJS | VietTQ | 14/01/2025 |
| Java Spring Boot | RuanNV | 15/01/2025 |
| Android Studio | HaLBH,HaLB | 20/01/2025 |
| Github | All members | 15/01/2025 |
| Figma | All members | 16/01/2025 |

## 3. Responsibility Assignments



**RACI Chart**: R~Responsible, A~Accountable, C~Consulted, I~Informed

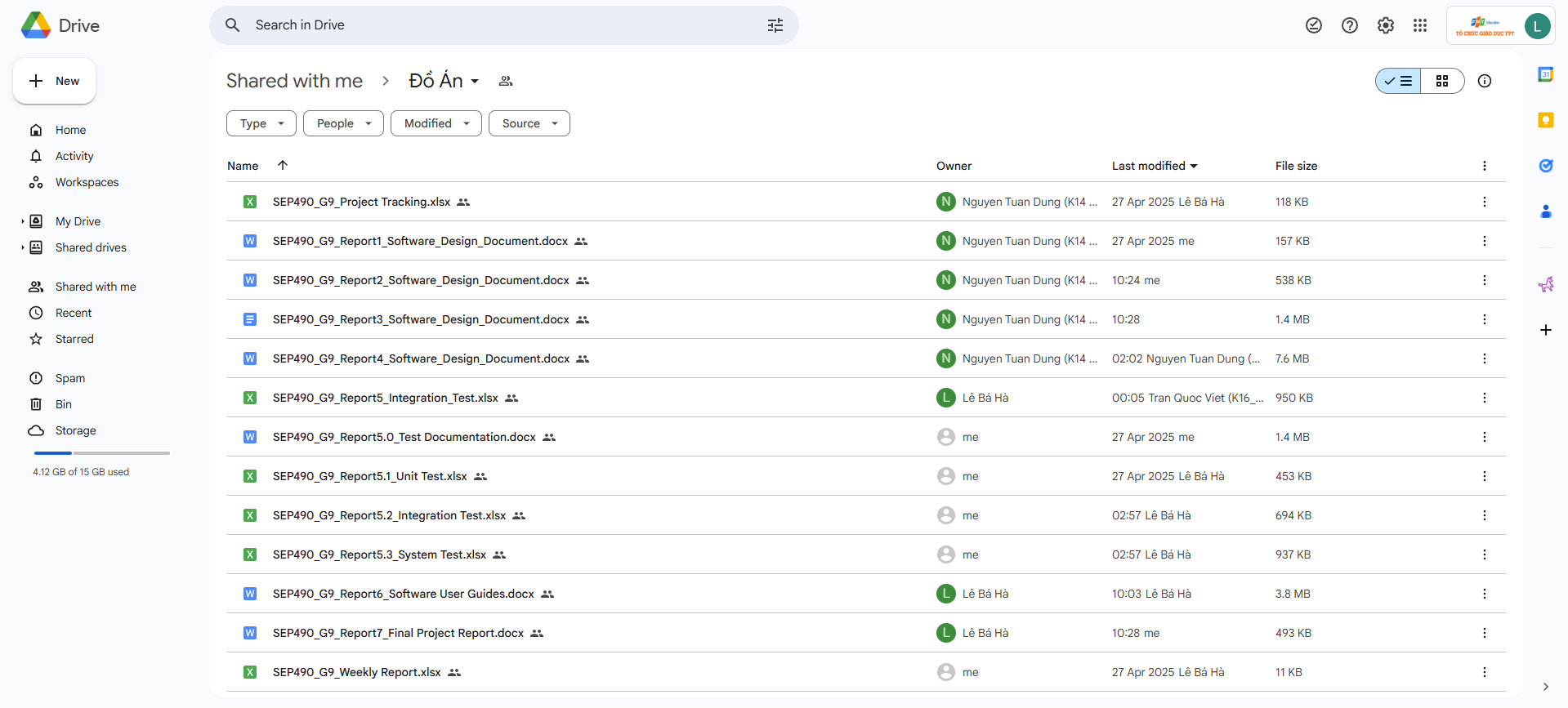
| Responsibility | HaLBH | HaLB | RuanNV | VietTQ | DungNT |
| --- | --- | --- | --- | --- | --- |
| Document and Report | | | | | |
| Project Introduction Document | C | I | R | R | R |
| Project Management Plan | C | A | I | R | I |
| SRS Document | R | D | A | A | R |
| SDS Document | I | R | A | A | R |
| Software Testing Documentation | C | R | C | A | C |
| Software User Guide | A | R | R | C | I |
| Final Report | R | A | R | C | I |
| Presentation Slide | R | C | R | I | A |
| Analysis and Design | | | | | |
| System Architecture Design | R | A | I | A | R |
| Use case Diagram | A | I | C | A | R |
| Class Diagram | R | C | I | R | I |
| Sequence Diagram | C | R | I | I | R |
| Main Workflow | A |  | R | A | C |
| UI Design |  | C | A | R | R |
| Package Diagram | R | R | I | C | R |
| Implementation | | | | | |
| Decide technique and tools | R | C | R | I | A |
| Tracking of development work | R | A | C | R | I |
| Create coding convention |  |  |  | A |  |
| Code back-end | C | A |  |  |  |
| Code front-end |  | C |  |  |  |
| Control source code | C | R | I | A | R |
| Deploy web to host server | I | R | R | C | A |
| Test | | | | | |
| Create Test Plan | C | R | A | I | R |
| Controlling testing activities | R | A | C | R | I |
| Create test report | R | A | I | A | C |
| Create unit test and bug log | A | R | R | C | I |
| Integration test | R | C | I | A | R |
| System test | A | R | C | I | A |
| Execute test | R | C | R | A | I |
| Report test result | C | A | R | R | A |

## 4. Project Communications

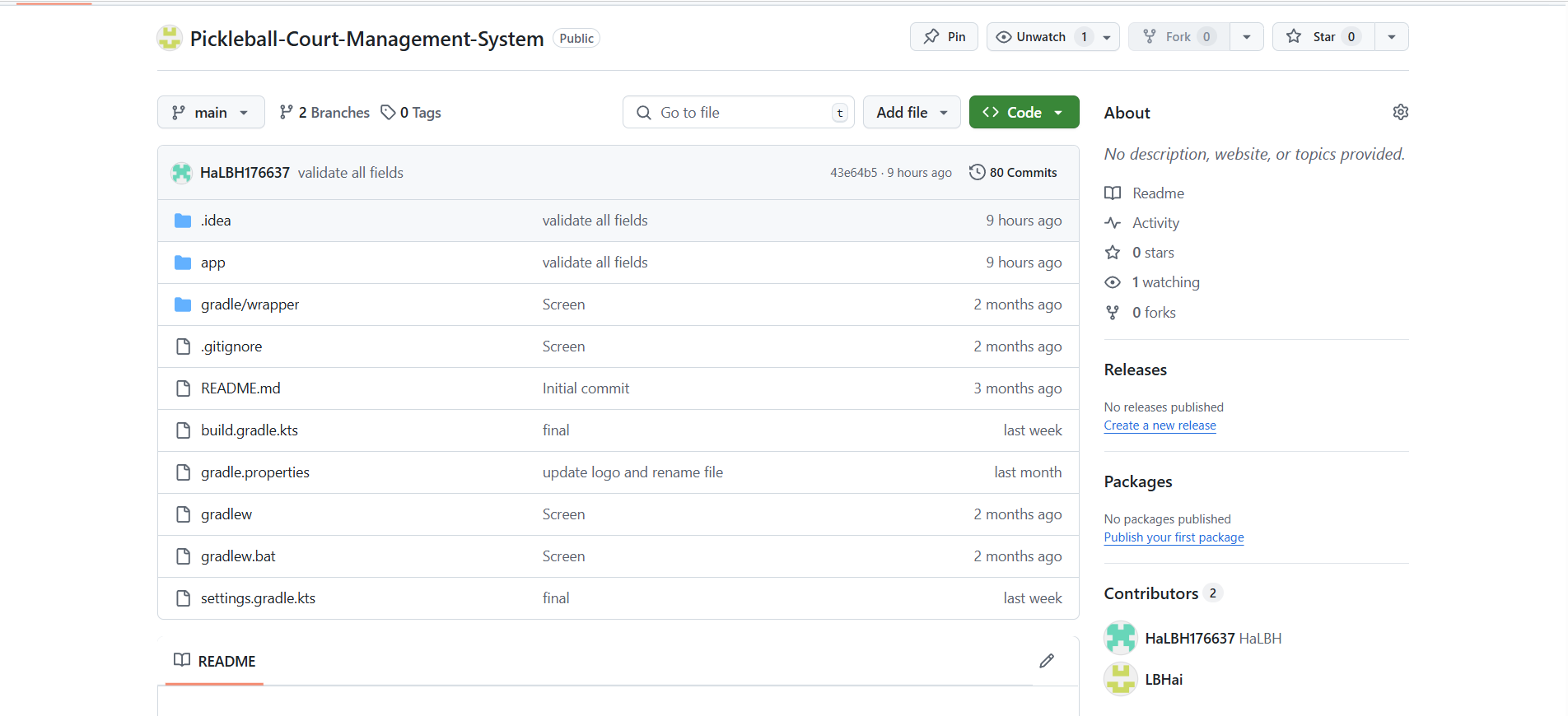
| **Communication Item** | **Who/ Target** | **Purpose** | **When, Frequency** | **Type, Tool, Method(s)** |
| --- | --- | --- | --- | --- |
| Meeting with mentor on a weekly basis | All team  member | Monitor on the project's progress and tackle issues. | 2 times per week | Online |
| Daily meeting | All team  members | Explore and clarify the partner’s requirements | 2 days per week | Google Meet |
| Daily Report | All team member | Report completed work. | Whenever someone finishes | Messenger, Zalo |
| Unscheduled meeting | All team members | Discuss and solve important issues | When there are issues of disagreement that need to be resolved and understood | Google Meet, Messenger, Zalo |

## 5. Configuration Management

### 5.1 Document Management

To manage our project documentation effectively, we utilize **Docs Online, Sheet Online**, and **Canva**, which enable us to track progress and document changes seamlessly. These tools help us organize **Document Reports, User Guides, Test Case Lists, and Slide Presentations**, making it easy to compare versions and consolidate updates. Using these applications ensures that all team members stay aligned, with clear and accessible documentation, allowing everyone to follow the project’s development with ease.  
  


### 5.2 Source Code Management

We use **GitLab** and **GitHub** to manage our source code, simplifying the process of storing, sharing, and updating code. These platforms allow us to assign tasks, collaborate efficiently, and manage changes quickly and accurately. This streamlined approach enhances our ability to deliver high-quality code for the **Pickleball Court Management System**, ensuring smooth development of features like court status tracking and payment automation.  
  
  


### 5.3 Tools & Infrastructures

### 

| **Category** | **Tools / Infrastructure** |
| --- | --- |
| **Technology** | Android Studio, Redis, Spring boot , Kafka , Spring security 6 |
| **Database** | SQL 2022, MySQL |
| **IDEs/Editors** | JRE 17 |
| **Diagramming** | DrawIO, Visual Paradigm |
| **Documentation** | Google Docs, Google Sheets, Canva |
| **Version Control** | Google Sheet, GitLab (Tasks, Defects, Issue Management) |
| **Deployment server** | VPS server |
| **Project management** | Google Sheet, GitLab (Tasks, Defects, Issue Management), Github , Jira |