Fpt University

TRƯỜNG ĐẠI HỌC FPT

CAPSTONE PROJECT REPORT

Report 2 – Project Management Plan

– Hanoi, August 2025–

Table of Contents

I. Record of Changes

Date	A*, M, D	In charge	Change Description
21/09/2025	A, M	DucNV	Add Project Overview, Configuration Management, Project Communications
27/07/2025	A	TuanLQ	Add Project Overview and Context Diagram Add
28/07/2025	A	SangNM	Add Responsibility Assignments Add
28/07/2025	A	VietDQ	Project Communications Add
28/07/2025	A	CuongNTT	Project Communications Modified, Add
30/07/2025	M	DucNV	Scope & Estimation Modified, Add
01/08/2025	A, M	DucNV	Project process Modified, Add

*A - Added M - Modified D - Deleted

II. Project Management Plan

1. Overview

1.1 Scope & Estimation

This section provides a structured view of a project's scope, breaking it down into manageable Work Breakdown Structure (WBS) items. Each WBS item represents a specific task or User Story within the project. For enhanced planning and resource allocation, complexity levels (such as simple, medium, and complex) are assigned to these items. Additionally, the estimated effort is quantified in man-days, indicating the anticipated number of working days required for each person to complete each task. Depending on the complexity and the amount of time taken on each task, we have defined 3 levels: simple, medium, and complex.

The definition of each level:

- **Simple:** Tasks that are straightforward, less complex, and require minimal resources and time to complete.
- **Medium:** Tasks with a moderate level of complexity, requiring a reasonable amount of resources and time to complete.
- **Complex:** Complex tasks that require significant resources and time to complete, potentially involving multiple steps and coordination between teams.

1.2 Project Objectives

We define one man-day as 8 hours. Our team works 8 hours per day from Monday to Friday, and 4 hours per day on Saturday and Sunday.

- The average working hours for one person per week is 48 hours (6 man-days).
- Total team capacity: 6 (man-days) × 14 (weeks) × 5 (members) = **420 man-days**.

#	WBS Item	Complexity	Est. Effort (man-days)
1	**Sprint 1: 06/09/2025 - 19/09/2025**		**70**
1.1	**Project Initiating**		**22.5**
1.1.1	Sprint planning meeting	Simple	0.5
1.1.2	Define project scope and objectives	Medium	5
1.1.3	Identify stakeholders and requirements	Complex	15
1.1.4	Create project charter	Simple	1
1.1.5	Define project team roles and responsibilities	Simple	1
1.2	**Document**		**6**
1.2.1	Prepare functional and customer requirements confirmation documents	Simple	2
1.2.2	Prepare Project Management Plan	Complex	4
1.3	**Design**		**36**

1.3.1	Build the web design system	Simple	2
1.3.2	Design web prototypes	Complex	12
1.3.3	Design database	Medium	8
1.3.4	Setup DevOps Google Cloud	Complex	14
1.4	**Testing**		**5**
1.4.1	Prepare the test plan	Medium	5
1.5	**Sprint Retrospective**		**0.5**
2	**Sprint 2: 25/09/2025 - 14/10/2025**		***70.5***
2.1	Organizing a meeting to confirm the functions and project requirements		1
2.2	**Document**		**22**
2.2.1	Prepare the SDS document	Complex	5.5
2.2.2	Prepare the SRS document	Complex	16.5
2.3	**Design**		**31.5**
2.3.1	Finish Web Prototypes	Medium	5.5
2.3.2	Draw UCD	Medium	5.5
2.3.3	Draw ERD	Medium	11
2.3.4	Draw Package Diagram	Medium	5.5
2.3.5	Define user flow	Medium	2.5
2.3.6	Update database design	Medium	1.5
2.4	**Coding**		**12**
2.4.1	Build the code base for the back-end user system	Medium	3
2.4.2	Build the code base for the back-end admin system	Medium	3
2.4.3	Build the code base for the front-end user system	Medium	3
2.4.4	Build the code base for the front-end admin system	Medium	3
2.5	**Testing**		**3**
2.5.1	Write UAE test cases	Medium	1.5
2.5.2	Write ST test cases	Medium	1.5
2.6	**Sprint Retrospective**		**1**
3	**Sprint 3: 15/10/2025 - 01/11/2025**		**39**

3.1	**Organize a meeting to confirm the project design documents and specifications**		**1**
3.2	**Document**		**12**
3.2.1	Update the requirements for the SDS document by comments	Complex	6
3.2.2	Update the requirements for the SRS document by comments	Complex	6
3.3	**Design**		**7.5**
3.3.1	Update diagrams by comments	Medium	5
3.3.2	Update Web Prototypes	Medium	2.5
3.4	**Coding**		**15.5**
3.4.1	Implement EPIC 1: Authentication for users(cont.)	Medium	6
3.4.2	Implement EPIC 1: Authentication for admin(cont.)	Medium	6
3.4.3	Build shared components for user web app	Medium	1.5
3.4.4	Build shared components for Admin web app	Medium	2
3.5	**Testing**		**4**
3.5.1	Write UT test cases	Medium	2
3.5.1	Write UT test cases	Medium	2
3.6	**Sprint Retrospective**		**1**
4	**Sprint 4: 03/03/2025 - 16/03/2025**		**58.5**
4.1	**Document**		**9.5**
4.1.1	Sprint planning	Simple	0.5
4.1.2	Finish updating SDS document	Complex	7
4.1.3	Finish updating SRS document	Complex	2
4.2	**Design**		**4**
4.2.1	Finish updating diagrams	Complex	2
4.2.2	Finish updating Web Prototypes	Medium	2
4.3	**Development**		**26.5**
4.3.1	Implement user authentication & session management	Medium	1.5
4.3.2	Implement user profile & address management	Complex	10
4.3.4	Implement admin user management (view, ban, unban)	Complex	10
4.3.6	Implement basic user interface pages	Simple	2

4.3.7	Implement notification feature (send & view notifications)	Medium	3
4.4	**Testing**		**8**
4.4.1	Unit testing for authentication & user features	Simple	2
4.4.2	Integration testing for core user flows	Simple	2
4.4.3	System testing for user interface	Simple	4
4.5	**Handover meeting – MVP phase 1**		**1**
4.6	**Sprint retrospective**		**0.5**
5	**Sprint 5: 17/03/2025 – 30/03/2025**		**83**
5.2	**Requirement & Design**		**1**
5.3	**Development**		**55.5**
5.3.1	Implement pet/profile management for users	Complex	6
5.3.2	Implement user preferences & attribute selection	Complex	6
5.3.3	Implement matching & interaction between users	Complex	6
5.3.4	Improve notification feature (read, unread status)	Medium	4
5.3.5	Implement report & complaint feature	Complex	6
5.3.6	Implement payment & membership feature	Medium	1.5
5.3.7	Implement admin content management	Medium	1.5
5.3.8	Complete user workflow based on pets & matching	Complex	7
5.3.9	Complete profile & matching user interface	Simple	1.5
5.3.10	Implement appointment & history management	Complex	7
5.3.11	Implement user activity statistics & badges	Complex	7
5.3.12	Implement admin attendance & monitoring features	Medium	4
5.4	**Testing**		**25**
5.4.1	Unit testing for user management features	Simple	6
5.4.2	Integration testing for end-to-end user flows	Simple	3
5.4.3	System testing for admin functions	Simple	2
5.4.4	System testing for end-user functions	Simple	4
5.4.5	Unit testing for payment & notification	Medium	6
5.4.6	Integration testing for payment processing	Simple	1
5.4.7	Final system testing for admin & user roles	Simple	3

5.5	**Handover meeting – MVP phase 2**		**1**
5.6	**Sprint retrospective**		**0.5**
6	**Sprint 6: 31/03/2025 – 13/04/2025**		**68.5**
6.1	**Documentation**		**7**
6.1.1	Update design documents for communication features	Medium	1
6.1.2	Prepare sequence diagrams for chat & consultation	Medium	2
6.1.3	Prepare class diagrams	Simple	2
6.1.4	Complete web user guide	Medium	2
6.2	**Development**		**52**
6.2.1	Implement user-to-user communication	Medium	4.5
6.2.2	Implement consultation between user and expert	Complex	6
6.2.3	Implement AI-assisted chat support	Complex	6
6.2.4	Implement expert confirmation & approval workflow	Complex	6
6.2.5	Implement blocking, reporting & usage limitation	Complex	6
6.2.6	Implement admin notification management	Complex	6
6.2.7	Finish updating Web Prototypes	Complex	6
6.2.8	Implement advanced admin user control	Medium	1
6.2.9	Complete user chat interface	Simple	1
6.2.10	Complete expert chat interface	Medium	2
6.2.11	Complete AI chat interface	Medium	2
6.2.12	Complete notification interface	Medium	1.5
6.2.13	Complete admin content management interface	Medium	2
6.2.14	Finalize admin support tools	Medium	2
6.3	**Testing**		**7**
6.3.1	Unit testing for communication features	Simple	2
6.3.2	Integration testing for chat workflows	Simple	1
6.3.3	System testing (user & admin)	Simple	2
6.3.4	Heuristic usability evaluation	Simple	1
6.3.5	Prepare acceptance testing document	Simple	1
6.4	**Handover meeting – MVP phase 3**		**1**

6.5	**Sprint retrospective**		**0.5**
7	**Sprint 7: 14/04/2025 – 27/04/2025**		**32.5**
7.1	**Requirement & Design**		**12.5**
7.1.1	Finalize web user guide	Simple	2
7.1.2	Update WBS & traceability documents	Simple	0.5
7.1.3	Complete all required project documents	Simple	10
7.2	**Development**		**4**
7.2.1	Fix user-side bugs & usability issues	Simple	2
7.2.2	Fix admin-side bugs & logic issues	Simple	2
7.3	**Testing**		**4**
7.3.1	Review & finalize UT, IT, ST, AT documents	Simple	4
7.4	**Final handover meeting**		**1**
7.5	**Closing Stage**		**11**
7.5.1	System handover & user training	Medium	1
7.5.2	Prepare & rehearse capstone presentation	Medium	10
Total Estimated Effort (man-days)			***421.5***

1.2.1 Test Phase Results Overview

#	Testing Stage	Test Coverage	No. of Defects	% of Defect	Notes
1	Unit Test	95%	0	0%	Written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended.
2	Integration Test	100%	3	2.8%	Individual software modules are combined and tested as a group.
3	System Test	100%	0	0%	Performed on the entire system, tests not only the design but also the behavior and even the customer's expectations.
4	Acceptance Test	100%	8	27.59%	System users perform tests in line with what

					would occur in real-life scenarios.

1.2.2 Ensure implementation on schedule with allowable resources

- Sprint Timeliness Rate(%): 90%
- Allocated Effort: 3024 hours - 420 man-days

#	Members	Weekdays	Weekends
1	Nguyen Van Duc	6 hours	8 hours
2	Nguyen Minh Sang	6 hours	8 hours
3	Nguyen Ta Tri Cuong	6 hours	8 hours
4	Le Quoc Tuan	6 hours	8 hours
5	Doan Quoc Viet	6 hours	8 hours

1.3 Project Risks

#	Risk Description	Impact	Possibility	Response Plans
1	Team member injury or illness	Very High	Low	Activate emergency response: reassign tasks, reduce sprint workload, and support affected members.
2	Team members lacking time due to academic commitments	High	High	Set a clear time commitment from the start, daily progress updates, and ensure minimum working hours.
3	Incorrect cost estimation (Cloud, domain, ID photo storage, etc.)	Medium	High	Prepare a detailed budget, review regularly, and use free GCP services where possible.
4	Technical difficulties with .NET, React, GCP, etc.	Low	Medium	Internal knowledge sharing, self-study with official docs, and peer support within the team.
5	Loss of user data or system failure	Very High	Low	Set up automated backups, and regularly test backup/restore functions to ensure data safety.
6	API have error	Very High	Medium	Ask the instructor for advice

2. Management Approach

The Pawnder project will follow the Agile Scrum methodology, ensuring flexibility, transparency, and continuous improvement. Development will be organized into short sprints, with each sprint delivering functional features such as cat profiles, matchmaking, AI-based predictions, and social sharing.

Regular Scrum ceremonies—planning, daily stand-ups, reviews, and retrospectives—will promote strong collaboration among developers, designers, AI specialists, and product owners. This approach enables quick adaptation to feedback, early delivery of value, and consistent alignment with user needs.

2.1 Project Process

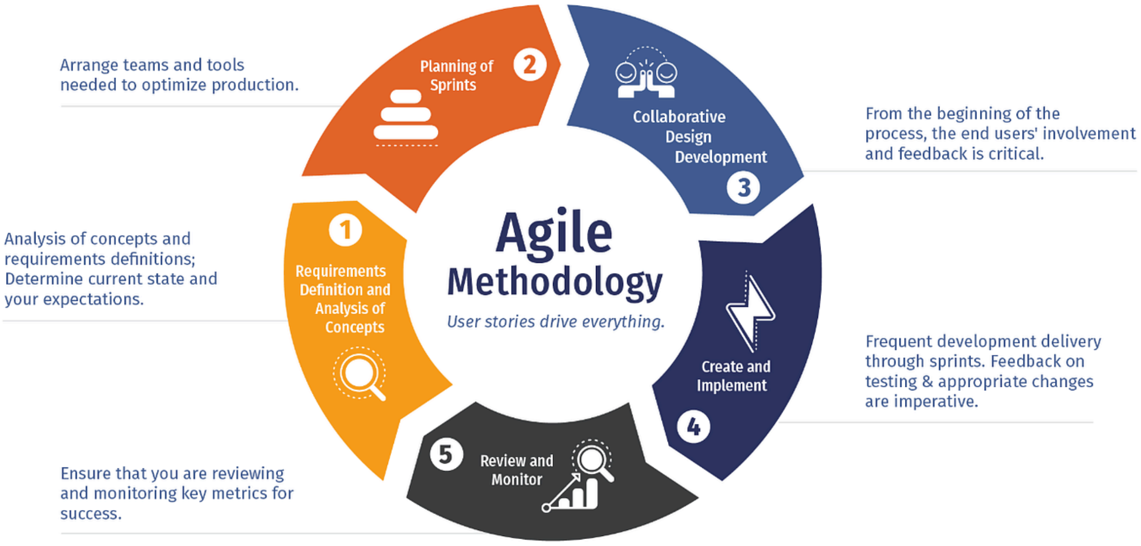

Agile
Methodology
User stories drive everything.
1
Requirements Definition and Analysis of Concepts
2
Planning of Sprints
3
Collaborative Design Development
4
Create and Implement
5
Review and Monitor
Arrange teams and tools needed to optimize production.
Analysis of concepts and requirements definitions; Determine current state and your expectations.
Ensure that you are reviewing and monitoring key metrics for success.
From the beginning of the process, the end users' involvement and feedback is critical.
Frequent development delivery through sprints. Feedback on testing & appropriate changes are imperative.

Figure 2.1: Agile Scrum methodology

The project is developed following the Agile Scrum methodology, which emphasizes flexibility, collaboration, and continuous improvement throughout the software development lifecycle. This approach allows the team to adapt quickly to changing requirements while ensuring consistent delivery of value to end users.

The process begins with Requirements Definition and Analysis, where business needs and user expectations are clearly identified and translated into user stories. Based on these requirements, the team conducts Sprint Planning to organize tasks, allocate resources, and define sprint goals.

Development is carried out through collaborative design and development, ensuring close cooperation among team members and active involvement from stakeholders. Features are then created and implemented incrementally within short, time-boxed iterations called sprints, typically lasting two weeks. Each sprint results in a potentially shippable product increment.

Finally, the team continuously reviews and monitors progress, evaluating performance metrics and gathering feedback to improve product quality and team efficiency. This iterative process ensures continuous enhancement, transparency, and alignment with user needs.

2.2 Quality Management

To ensure high-quality delivery of the **Pawnder** project, the team will apply a structured quality management approach throughout the development lifecycle. Quality objectives will be achieved through the following practices:

- **Defect Prevention**: The team will follow coding standards, naming conventions, and best practices to minimize defects from the start. Regular knowledge-sharing sessions and documentation will help reduce misunderstandings and improve consistency.
- **Reviewing**: Code reviews and design walkthroughs will be conducted at each sprint to identify issues early. This process also ensures alignment with requirements and promotes knowledge transfer within the team.
- **Unit Testing**: Each module, such as profile creation, matchmaking, and AI prediction, will undergo unit testing. This ensures components work correctly in isolation before moving to the next stage.
- **Integration Testing**: After unit testing, the system will validate how modules interact, focusing on data flow and functionality across features. This step ensures the app functions smoothly as a cohesive platform.
- **System Testing**: End-to-end testing will be performed to verify performance, usability, and reliability from the user's perspective. This guarantees that the final product meets both functional and non-functional requirements before release.

2.3 Training Plan

Training Area	Participants	When, Duration	Waiver Criteria
.NET	All member	Week 1 – 2 sessions (2 hours each)	Mandatory
React Native	All member	Week 2 – 3 sessions (2 hours each)	Mandatory
Git, Github	All member	Week 1 – 1 session (2.5 hours)	Mandatory
Visual Paradigm	All member	Week 2 – 3 sessions (1.5 hours each)	Mandatory
Testing (Unit & UAT)	All member	Week 4 – 2 sessions (2 hours each)	Mandatory

3. Project Deliverables

#	Deliverable	Due Date	Notes
1	SEP490_G151_Report1_ProjectIntroduction.docx	15/09/2025	Project introduction
2	SEP490_G151_Report2_ProjectManagementPlan.docx	22/09/2025	Project Management Plan
3	SEP490_G151_Report3_SoftwareRequirementSpecification.docx	06/10/2025	Software Requirement Specification (SRS)

4	SEP490_G151_Report4_SoftwareDesignSpecification.docx	20/10/2025	Software Design Specification (SDS)
5	SEP490_G151_Report5_TestDocumentation.docx	10/11/2025	Test Documentation
6	SEP490_G151_Report6_SoftwareUserGuides.docx	24/11/2025	Software User Guides
7	SEP490_G151_Report7_FinalProjectReport.docx	15/12/2025	Final Project Report

4. Responsibility Assignments

D~Do; R~Review; S~Support; I~Informed; <blank>- Omitted

Responsibility	SangNM	DucNV	CuongNTT	TuanLQ	VietDQ
Project Planning & Tracking	S	D	R	R	S
Prepare Project Introduction Document	S	S	D	R	S
Prepare SRS Document (Overview Part)	R	D	S	S	S
Prepare SRS Document (User Requirements)	D	R	S	S	S
Prepare SRS Document (Functional Requirements)	D	S	R	S	S
Prepare SRS Document (Non-Functional Requirements)	R	D	S	S	S
Prepare SRS Document (Use Case Diagrams)	S	R	D	S	S
Prepare SRS Document (Activity Diagrams)	R	S	D	S	S
Prepare SRS Document (System Models)	S	R	S	D	S

Prepare SDD Document (Architecture Design)	R	D	S	S	S
Prepare SDD Document (Database Design)	S	R	D	S	S
Prepare SDD Document (UI/UX Design)	R	S	D	S	S
Frontend	S	R	R	D	D
Backend	D	D	D	R	R
Integration & Testing	S	R	S	D	S
Deployment	R	S	D	S	S
Prepare Final Report	D	R	S	S	R
Prepare Final Presentation	R	S	D	S	R

5. Project Communications

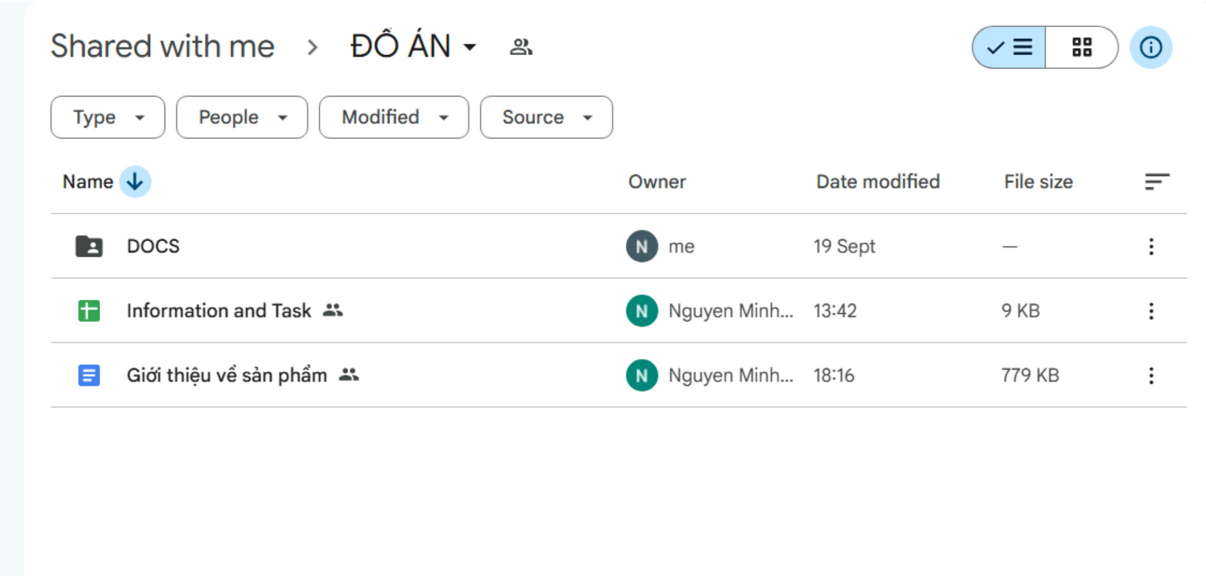
Communication Item	Who/ Target	Purpose	When, Frequency	Type, Tool, Method(s)
Daily Stand-up Meeting	Entire team (developers, testers, product owner)	Update progress, share issues, and assign daily tasks	Daily, 15 minutes (every morning)	Face-to-face / Google Meet; task notes recorded in Jira
Sprint Planning Meeting	Product owner, Scrum master, development team	Define sprint goals, select and estimate backlog items	At the beginning of each sprint	Online meeting (Google Meet); backlog managed in Jira
Sprint Review Meeting	Development team, product owner, stakeholders	Demonstrate completed features and collect feedback	At the end of each sprint	Online presentation; feedback documented in Jira
Sprint Retrospective	Development team, Scrum master	Evaluate sprint performance and identify improvements	At the end of each sprint	Team discussion via Google Meet; action items recorded

Backlog Refinement	Product owner, development team	Clarify and prioritize product backlog items	Once per sprint (or as needed)	Jira backlog review; online discussion
Technical Discussion / Design Meeting	Developers, technical lead	Discuss system design and technical solutions	As needed	Google Meet / Chat; diagrams on Draw.io

6. Configuration Management

6.1 Document Management

We use Google Drive to manage project documents and their changes/versions, such as the image below:



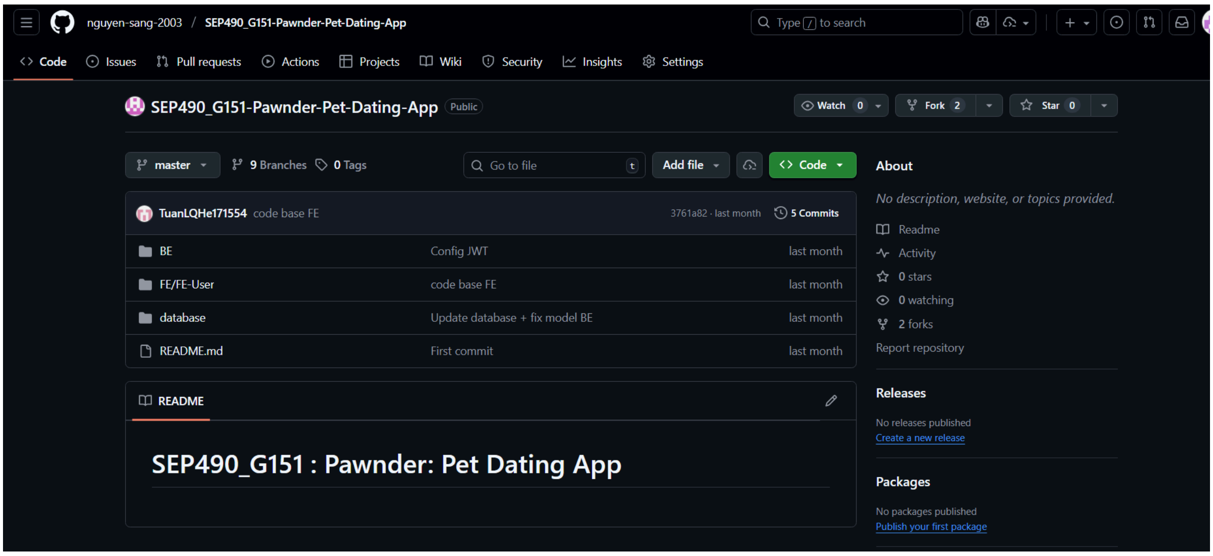
Name	Owner	Date modified	File size
DOCS	me	19 Sept	—
Information and Task	Nguyen Minh...	13:42	9 KB
Giới thiệu về sản phẩm	Nguyen Minh...	18:16	779 KB

Figure 6.1: Document Management

Changes are tracked and different versions are managed using Google Drive, which allows us to manage project documents and their changes/versions efficiently. Each document's history is automatically saved, enabling us to view and restore previous versions if needed. This ensures that all changes are documented, and different versions are easily accessible for review and comparison.

6.2 Source Code Management

The source code will be managed by the team on GitHub:



SEP490_G151-Pawnder-Pet-Dating-App

TuanLQHe171554 code base FE 3761a82 · last month 5 Commits

BE	Config JWT	last month
FE/FE-User	code base FE	last month
database	Update database + fix model BE	last month
README.md	First commit	last month

SEP490_G151 : Pawnder: Pet Dating App

About

No description, website, or topics provided.

Readme
Activity
0 stars
0 watching
2 forks
Report repository

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Figure 6.2: Source Code Management

For the branching strategy, we use the Gitflow branching strategy. This involves creating separate branches for features, releases, and hotfixes.

Code reviews will be conducted using pull requests. When a developer completes a feature or bug fix, they will create a pull request to merge their changes into the develop branch. The pull request will be reviewed by at least one other team member, who will check for code quality, adherence to coding standards, and potential issues. Feedback will be provided, and changes may be requested before the pull request is approved and merged.

The release management process involves the following steps:

- **Release Planning:** Identify the features and fixes to be included in the release.
- **Creating a Release Branch:** Create a release branch from the develop branch.
- **Final Testing and Bug Fixes:** Perform thorough testing and address any issues found.
- **Release Approval:** Obtain approval from stakeholders for the release.
- **Merging and Tagging:** Merge the release branch into the master branch and tag the release with a version number.
- **Deployment:** Deploy the new release to the production environment.
- **Post-Release Monitoring:** Monitor the release for any issues and address them as needed.

6.3 Tools & Infrastructures

Category	Tools / Infrastructure
Technology	React (FrontEnd), React Native(App), DotNet(BackEnd)
Database	PostgreSQL, FireBase
IDEs/Editors	Visual Studio Code, VS Code
Diagramming	DrawIO, Visual paradigm
Documentation	Ms Office, Google Docs/Sheets/Slides
Version Control	GitHub(Source Codes), Google Drive (Documents)
Deployment server	Azuze cloud

Project management	Jira, Google Sheets