FIT3170
Software Engineering Practice

Milestone 4
Maintenance Plan

HansRoslinger

Yubi

Yubi

Table of contents

Overview

As the unit concludes and the team is unlikely to continue collaborating on this project, this document outlines our plan for its future maintenance. Individual team members may choose to fork the project repository and continue its development independently after the semester ends. However, the main repository is unlikely to receive further updates unless someone takes on the responsibility of maintaining it.

The team is currently in discussions with the Monash DeepNeuron student team, who are considering taking over the maintenance of the repository. However, this arrangement has not yet been confirmed. If they decide not to assume this responsibility, the project's life cycle will unfortunately come to an end.

Purpose

This document serves as a guide for future developers who wish to maintain the project, whether they are part of Monash DeepNeuron or external contributors interested in maintaining, updating, or further developing the application. It outlines the key processes, tools, and best practices required to ensure the project remains functional, stable, and secure over time. Additionally, the document provides an overview of the existing documentation created for the project and indicates when and how each resource should be referenced during the maintenance phase.

The plan also outlines contingency measures in the event that no individual or team is available to take over maintenance. This includes guidance on handling hosting, user assets, and database management should the project become inactive.

Future Maintenance

If Monash DeepNeuron or another party decides to take over project maintenance, the following plan outlines the key steps and responsibility required to have a smooth transition and continued development. This section is intended to provide clear guidelines for future developers to keep the project running.

Future developers are responsible to:

- Keep the application running smoothly and securely on modern infrastructure.
- Maintain affordable hosting and storage costs.
- Support handovers to other developers with minimal friction and no knowledge loss.
- Document key processes so future contributors can get started quickly.

- Keep a decision record to list any major decision on the structure and architecture of the project.

Roles & Responsibility

To maintain the project effectively, the following roles should be assigned. In smaller teams, one individual may handle multiple responsibilities:

- **Product Lead:** manages priorities, tasks, and communication with supervisors or stakeholders.
- **Tech Lead:** oversees architecture, reviews schema or code changes, and maintains system health.
- **Infrastructure Manager:** maintains access to Vercel, AWS, and GitHub; handles backups and secrets.
- **Documentation Owner:** keeps README, setup guides, and changelogs updated.

Existing Project Documentation

To support future maintainers and developers, several documents have already been created and stored within the `/documentation` directory of the repository. These resources should serve as the foundation for understanding the project structure, setups, privacy policy and operational procedures. Each document should be kept up to date.

- REAMDE.md: Provide an overview of the project, repository layout, setup instructions, deployment process and common issues faced during the setup process. It should be the starting oint for any one new to the project
- project_roadmap.md: Outlines completed features, ongoing development and planned future improvements and features. Maintainers should update this document at the end of each release cycle.
- branching_versioning_strategy.md: Details the Git workflow, branch naming convention, commit message format and pull request guidelines. This document should be referred to when making contributions to the project. This ensures consistent collaboration and clean version strategy.
- maintenance_plan.pdf: This document explains roles, processes, access control, and continuity planning for future developers or teams.
- user_guide.pdf: Provides instruction for end user on how to use and operate the applications and its key features
- privacy_policy.pdf: Describe how the team manages data. This document must be reviewed before making any changes that involve personal data, authentication or assets uploads. This should be updated if any changes to the privacy policy is made.
- known_bugs.md: List the known system limitations and unresolved issues.

Core Maintenance Practices

New maintainers should:

- Follow the branching and versioning strategy outlined in the documentation.
- Review and update all dependencies, ensuring compatibility with the latest stable releases.
- Maintain documentation consistency in the `/documentation` directory, ensuring it remains the single source of truth.
- Keep the CI/CD pipeline active through GitHub and Vercel integrations.
- Regularly test critical features such as authentication, gesture detection, and media handling.

General Guidelines

Development and Deployment

- Work in short-lived feature branches and merge through pull requests.
- Each PR must pass CI checks and at least one peer review.
- Merging to “main” triggers automatic production deployment through Vercel.
- Prisma migrations handle all database changes, no manual edits.
- Preview deployments are generated automatically for PRs.
- Rollbacks can be done by reverting the last merge and redeploying.

Access and Security

- Only the Infrastructure Manager holds admin privileges.
- Rotate API keys and passwords every year.
- Remove access immediately when a member leaves the team.

Tooling and Dependencies

- Standardize Node, TypeScript, and Prisma versions in documentation.
- Use Dependabot or Renovate for automated dependency updates.
- Run “npm audit” monthly and resolve all high-severity issues promptly.

Testing and CI/CD

- Run type-checking, linting, and unit tests in CI before merging.
- Smoke test the main user flows after each deployment.
- Only release when all checks pass and documentation is updated.

Monitoring and Backups

- Use Vercel analytics and uptime monitoring to detect outages.
- Store function logs in Vercel and AWS.
- Automate daily database snapshots and verify restore tests monthly.
- Apply lifecycle rules on AWS buckets to reduce storage costs over time.

End of release life cycle

At the end of each release life cycle, the maintainer should:

- Update documentation, decision records, and runbooks.
- Rotate credentials and confirm all access removals.
- Summarize open issues and the current roadmap.

Onboarding New Members

New developers should be onboarded to the project preferably by other developers in the team. But in the case of no one is available to onboard them, the following things should be done:

- Read the README, the documents created for the project and the architecture of the project.
- Set up the local environment and confirm the app runs correctly.
- Gain access to GitHub, Vercel, and AWS.
- Submit a small pull request within the first week to learn the workflow.

Risk Management

- Keep architecture diagrams and decision records updated to reduce key-person dependency.
- Test every database migration on staging before production deployment.
- Rotate credentials immediately if any secret is exposed.
- Periodically test monitoring alerts and rollback procedures.

Project Shutdown Plan

If no team or organization takes the responsibility for maintaining the project, the following plan outlines the steps required to safely decommission all systems, preserve project assets, and comply with data management and privacy standards.

Github Repository

- The repository will be set to read-only.
- Contributors are free to fork the repository and maintain the project.

Hosting and Deployment Services

- Allow all existing Vercel deployments to expire once the free-tier period concludes.
- Remove the associated project from the Vercel dashboard to prevent unauthorized redeployment.
- Revoke all Vercel tokens and environment variables stored within the repository or cloud platform.

Database and Storage

- **AWS S3:**
  - Delete all uploaded files and media assets stored in the project bucket.

    - Remove access policies and user credentials associated with the bucket.
    - Verify that no public access points remain.
- **Prisma / Database Service:**
    - Permanently delete all records from the database.
    - Disable and remove all database connections and credentials.

Documentation and Handover

- Ensure all key documents are finalized and stored in the `/documentation` directory.
- Include a short “Project Summary” section in the README or create another document outlining the purpose of the project, technology used, final version number and release date and reason for project closure.
- Include a summary of what would be required to relaunch the project in the future (e.g., restoring backups, redeploying on Vercel).