Software Development Process (SDP)

SENIOR SOFTWARE ENGIN PROJECT (CS_461_001_F2023)
Updated Farm Directory
Ehan Wentworth, Zhanhong Liang, Trenton Fisher

Principles

- Team members will respond to asynchronous communications within 24 hours, unless they get prior approval from the team for a longer response time (i.e. holidays/vacations).
- Team members will use Github Issues or Projects to keep track of the work.
- Team members will plan for at least one to two weeks ahead of time.
- All new features need to work in branches of the main repository.
- GitHub pull requests have to be reviewed by and get both other teammate's approval.
- The pull request will need to meet all criteria in the Definition of Done.

Process

For our software development process, we plan to adopt a waterfall approach with a small amount of elements from agile, this will best suit our part time asynchronous workflow and require continuous development and communication

- Backlog and Planning (1/week)
    - Once a week group members will meet and discuss the current backlog which will contain user stories, tasks, issues, and features.
    - Team members will use asynchronous communication tools like Slack, Discord and email so we can use threaded messages for discussions.
    - During this time we will refine the backlog issues, priorities and documentation platform to ensure it aligns with the clients expectations and make sure that all the work can be done on time.

- Kanban Board (To Do - In Progress - In Review - Done)
    - We will utilize GitHub Projects and Issues for asynchronous collaboration and code reviews and to maintain a board featuring tasks in the "To-Do" section, each with varying priorities. Team members will select a task, move it to the "To-Do" section, and, once completed, transition it to the "In Review" stage for a code review. Once the code is merged and finalized, it will be moved.
    - This will allow tracking of status and progress, continuous development, and organization
- Demo/Review with Stakeholders (1/every two weeks)
    - Once every two weeks during stakeholder meetings we will have a discussion of what we have done, if the changes are able to be demonstrated then we'll provide a demo, if not an in depth discussion will be had regarding any progress, questions, and roadblocks.

Roles

Team Leader: Ehan Wentworth
Team Member: Zhanhong Liang
Team Member: Trenton Fisher
Project Owner: Katie Meade
Project Manager(client): Jim Cupples

Tooling

Version Control	GitHub
Project Management	GitHub Issues and Projects
Documentation	https://github.com/withastro/starlight and README.
Test Framework	Playwright & Jest.
Linting and Formatting	Prettier
CI/CD	GitHub Actions
IDE	Visual Studio Code & NeoVim

Graphic Design	Figma, Pen and Paper, Google Slides & Draw.io
Others	AI assistants, creation of code executable, monitoring, code analysis, Word doc.

Definition of Done (DoD)

- Acceptance Criteria Validation: Ensures that the task or feature meets all specified requirements. This usually includes reviewing user stories or task descriptions.
- Code Quality and Reviews: Code meets the coding standards; reviewed both other members, all the feedback from code reviews addressed.
- Testing: All unit tests, integration tests and user stories are satisfied.
- Complete Implementation: If a feature or task spans multiple components such as backend, frontend, database, library, etc, all relevant parts will implement the changes consistently.
- Summary of changes updated for project partner meetings are demoed and approved.

Release Cycle

- Automatically deploy to staging every merge to the main branch.
- Deploy to production every release.
- Release every half term (3-4 weeks).
- Use semantic versioning MAJOR.minor.patch.
    - Increment the major version for new features / chief functionalities.
    - Increment the minor version for feature modifications.
    - Increment the patch version for bug fixes.

Until the API is stable, major should be 0.

Environments

Environment	Infrastructure	Deployment	What is it for?	Monitoring
Production	Vercel	Via CI/CD upon each major release with pull request	For real users accessing the directory and searching farms.	Vercel provides in depth built in monitoring through their dashboard
Staging (Test)	Vercel	Via CI/CD for each merge to	Testing new features, updates, and bug fixes.	Vercel

		the dev branch with pull request		
Dev	Local (macOS and Windows)	Manual commit	Development, unit tests and debugging.	Local unit tests passed

This environment structure will help in the scope of the project as no long continuous server side code needs to be run. Simple Oauth, one time search queries, and updating user profiles is all that is happening, so pay as you go serverless cloud providers will be the most efficient and cost effective methodology. Along with having integration tests, Vercel has advanced web analytics, and git branch deployments built in, this allows for ease of use under one dashboard with generous pricing.