Inlyne – Team Working Agreements & Definition of Done

Team Members: Katarina, Taiga, Alex, Jason, Jivaj, Manuel
Date: June 3, 2025

Team Working Agreements

Task Management

- Team members will volunteer for tasks and take ownership based on interest and expertise
- All task updates should be recorded and maintained in Jira.
- Team members are expected to complete a minimum of 10 story points per sprint.
- Every member will make a good-faith attempt to complete assigned tasks before the sprint ends
- Groups determined based on interest and maintain most responsibility for task that fall under their branch
  - Backend: Alex, Kat
  - VSCode Ext: Manuel, Jason
  - Web App: Taiga, Jivaj

Meetings & Attendance

- Weekly Meetings:
  - Monday @ 5:15 PM – Quick sync (30 mins): review progress, address blockers, and set weekly goals.
  - Wednesday @ 9:30 AM – Deep work session (1+ hour): focused collaborative coding.
- Pair Programming: Teams of 2 will meet independently throughout the week to tackle specific issues and report updates in the group chat.

- Attendance at all meetings is expected. If a meeting is missed, the team member must follow up with their mini-group or the Scrum Leader

Communication

- Primary Communication Tool: Team text group chat
- Communicate through the week in between meetings to make up for lack of daily standup meetings
- All updates, blockers, and decisions should be shared transparently with the team.

Workspace & Collaboration Tools

- Designated Work Room: Reserved library room for in-person collaboration.
- Project Repository: Hosted on GitHub, structured into three main sections:
  - VSCode Extension
  - Web Application
  - Backend Services

Development Environment

- IDE: VSCode (preferred by all team members).
- Platform: Local development with potential use of virtual machines or containers as needed.
- Coding Standards: (see Style Guide for more details)

Definition(s) of Done

1. WebApp (Frontend)

- UI matches design specs and functions correctly across major browsers
- User input is validated both on the client side and appropriate feedback is provided
- Component code is modular, well-documented, and follows team style guide
- No console errors or broken elements during manual testing.
- Feature has been peer reviewed and merged into the main branch.

2. Backend

- API endpoint is implemented, documented and passes integration tests.
- Database interactions are secure, validated, and do not break existing schema or functionality
- Endpoint handles all expected inputs and edge cases.
- Proper error handling and logging are implemented for failures.
- Endpoint is deployed and functional in the dev or staging environment.
- Changes are reviewed and merged into the main backend branch

3. VSCode Extension

- Feature is accessible via the command palette or UI and behaves as intended in VSCode.
- Communication between the extension and backend is tested and stable.
- Real-time updates sync properly between the editor and shared document view.
- Extension is tested on at least one supported OS (Windows, macOS, or Linux).
- README is updated with installation and usage instructions for the new feature.