Andrew Agile Platform

17-637 Web Application Development - Project Proposal

Team members

- Nianyi Guo nianyig@andrew.cmu.edu
- Yucen Xu yucenx@andrew.cmu.edu
- Zhiqi Li zhiqili@andrew.cmu.edu
- Peng Zhao pzhao2@andrew.cmu.edu

Project description

The Andrew Agile Platform provides a web-based, Kanban-style, list-view application, which helps teams to collaborate and manage project progress. The main page shows a board with sticky notes where tickets could be moved across different columns. The columns indicate project workflow and the sticky notes suggest the ticket ownership and status. It emphasizes the ease of use with a variety of ticket operations needed by users. This application is designed for all types of users who need a tool to improve productivity through their project management process. Some of the similar SaaS products on the market include Trello, Jira and Notion.

Ticket definition: Ticket is the major component of this website, which is the basic unit throughout the project progress.

- Type: We provide four types of tickets, including project, task, issue and action. Project is a high-level concept. Task refers to the plain, normal job. Issue is similar to bugs. And action is used for access management.
- Status: The task, issue and action tickets will have four status: backlog, todo, in progress and done.
- Attributes: Detailed information about tickets, including title, priority, label, assignee, reporter, and some advanced information.

Basic features:

- Kanban board:
  - View: There are four columns spread horizontally across the Kanban board, corresponding to four status. Within each column, tickets are represented as blocks that spread vertically.
  - Search and filter: Users can search tickets by keywords, and also filter the tickets on the assignee, reporter, priority or/and label.
  - Drag: Ticket blocks can be dragged among four columns to change their status.
- Ticket details: A ticket detail panel will pop up after clicking the ticket on the Kanban board. In that panel, the ticket's status, description and activities can be edited.

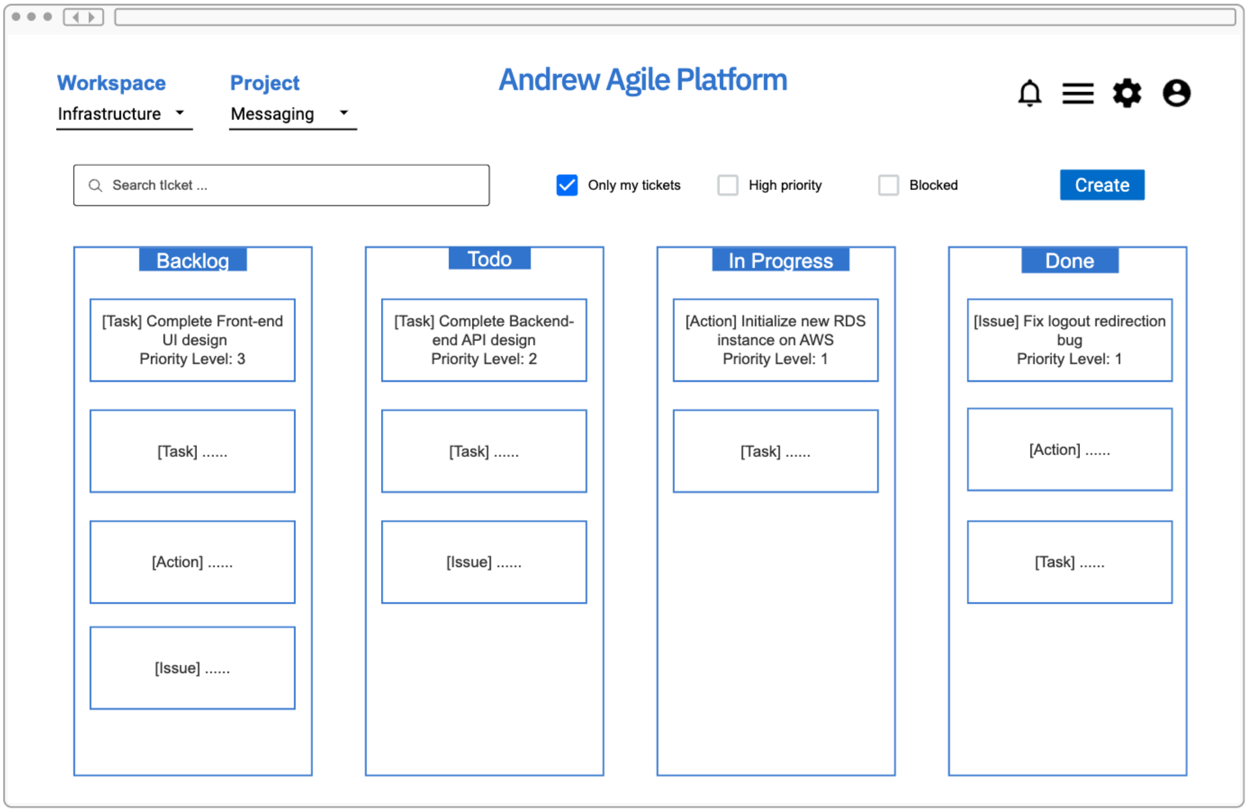
Intermediate features:

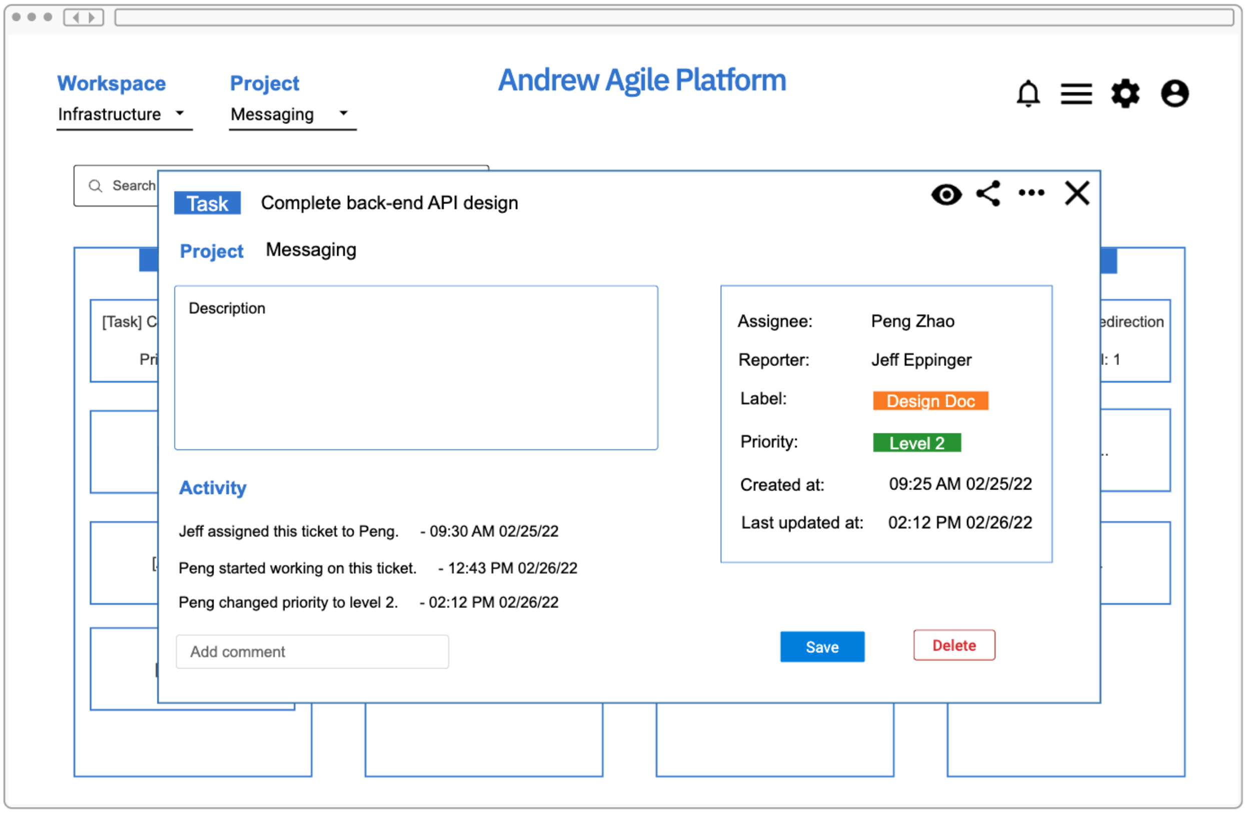
- Sprints: Users can define sprints and assign tickets to a certain sprint.
- Gantt chart view: A chart displaying the tickets against time, showing in which sprints tickets are completed and the timespan it takes to complete.
- Ticket relations: Tickets can be linked to each other by relationships including blocked by, caused by and sub-task.
- Agile statistics: The website provides a statistics board to see if the teams are working well under the agile method, by aggregating ticket completion data.

Advanced features:

- Workspaces and teams: Workspace is a higher-level abstraction for projects. Each team will have its own workspace with a list of projects. Therefore we have access control functionality.
- User access management: User will be assigned a role for each workspace/project he or she participates in. Different roles have different access to Kanban board and tickets (view/comment/edit).

UI prototype





Task Complete back-end API design

Project Messaging

Description

Assignee: Peng Zhao

Reporter: Jeff Eppinger

Label: Design Doc

Priority: Level 2

Created at: 09:25 AM 02/25/22

Last updated at: 02:12 PM 02/26/22

Activity

Jeff assigned this ticket to Peng. - 09:30 AM 02/25/22

Peng started working on this ticket. - 12:43 PM 02/26/22

Peng changed priority to level 2. - 02:12 PM 02/26/22

Add comment

Tech stack

As required by the project spec, we will use Django as our backend framework. Although Django does provide HTML templates to reduce the frontend workload, dynamic state binding is still a headache when it comes to many user interactions. Therefore we would like to try React because it provides a better state-binding, event-driven and component-based frontend framework. We would also like to adopt Material-UI to build simple and clean web pages under the material design paradigm.

In short, Django will simply act as the backend API server, so the controller functions in views.py will take care of business logic but only return JSON-based responses. And React components will listen for state changes, send Ajax requests to the API server, and then parse responses to update the states.

If things go wrong, for example, the efforts of learning React and MUI are too costly, we can switch back to the Django-based solution (just like the course homework), and use Bootstrap etc. to assist the CSS styling.