MALTE

Multi-access Live Text Editor

Progress Report 2

Team Name: #dream-team

I. Summary of Work

This is the first progress report with actual coding work done, as such, all code commited to master will be summarized here, together with a brief explanation of progress in terms of features.

A. Features and Changes

We have setup a website at https://rehnarama.github.io/MALTE/ where all project info can be found. We have setup up a GitHub project – `rehnarama/MALTE` – which will serve as the central repository for all code. The repository itself is structured as a monorepo which means that all code, even though they are logically separate projects, all reside within the same repository, e.g. the frontend and backend. In the following subsections we will summarize changes across all projects within the repository.

To ensure that no regressions occur during development we have set up a CI system using GitHub Actions and a CD system using Heroku. On every commit, our regression test suite will be run together with a strict linter. If any errors are reported, the pull request will be unavailable to merge into the master branch. This ensures that the master branch is always working. Whenever a pull request is merged to master, our CD system using Heroku will be triggered. Heroku will build our system and deploy it, meaning that there is an always up-to-date version of the system accessible on the address https://malte-uu.herokuapp.com/.

frontend: A boilerplate has been set up with three main areas: the file tree view, the terminal view, and the text editor view. The different areas can be resized with small handles between each section.

The file tree view can both render files and folders, allow for collapsing and expanding of folders. At last, the file tree view provides affordances to create a file or folder and delete a file or folder, providing that folder is empty.

The terminal view is at the bottom of the screen and can both print terminal output as well as accept terminal input. We utilise the library Xterm.js[1] for comprehensive VT support[2]. The terminal resizes dynamically according to the browser size.

The text editor displays a file which is accessed from the backend which can be edited by focusing the editors, and start typing. We utilise the Monaco editor[3] for syntax highlighting, auto-completion and IME support. Different from most editors is that the backing data in not a string, but a Replicable Growable Array (RGA), as written about in the project specification. To support this, internal Monaco operations had to be converted to RGA operations. In the text editor you can see text that other clients are writing, but you cannot edit the same file concurrently due to your caret losing its position.

At last, the frontend exposes a `Login with GitHub` button which can be used to authenticate oneself with GitHub. Using this functionality, your GitHub avatar will be shown floating in the top right corner. This will later be used for permission management.

backend: The backend project contains all necessary plumbing to support the features of the frontend. This includes a WebSocket server so that a duplex connection between the frontend and backend can be established. The backend also provides the GitHub integration discussed above; since the GitHub authentication is the basis of permission management, the backend must access the GitHub OAuth API to ensure that the client is who they claim to be.

Furthermore, the backend spawns and manages the lifetimes of pseudo terminals (PTY) to ensure that each client has their own terminal to use on the frontend, as well as closing this process when the client disconnects.

The backend also manages buffers with all open files in the system, as well as periodically saving the buffers to the file system so that no changes are lost, even in the event of a crash. The backend will later send changes from one client to all other connected clients, to enable the collaborative editing experience as explained in the project specification.

At last, the backend both sets up the project workspace as well as watching this workspace folder for any

[1] https://xtermjs.org/

[2] https://github.dyf62976.workers.dev/xtermjs/xterm.js/wiki/VT-features

[3] https://microsoft.github.io/monaco-editor/

changes. This is necesseary since files can be removed, moved or created using the terminal, and these changes must be reflected on the frontend file tree view.
rga: The RGA project currently contains the RGA data structure as discussed above and the project specification. It resides in a separate project since both the backend and frontend needs to have access to it. At a later time, the RGA project will also contain abstraction to maintain cursor positions when editing the document concurrently.
malte-common: The malte-common project contains common functionality between other projects, such as API type definitions which must be accessed by both backend and frontend, but is not substantial enough to require its own subproject as in the case of the RGA.

B. Changelog

A changelog that reflects the work from the beginning of the project until now can be found in this link to GitHub.

II. Difficulties

Setting up continuous proved to be difficult due to Cross-Origin Resource Sharing (CORS) troubles[4]. This was later solved but a considerable amount of time was spent solving it.

III. Time Plan

We are on track on our current schedule as seen in the GANTT chart in the project specification. We are even a bit ahead of schedule as the foundation of *User Account Management* has been laid with the GitHub integration.

IV. Next Sprint

During next sprint we will progress with the main functionality of the system, namely the collaborative editing experience. We also aim to have a notion of user accounts, permissions management and functionality supporting the collaborative experience, e.g. cursor synchronization and a list of all who are online. Next progress report deadline are in the middle of next sprint, as such, it is not expected that all of these functionality should be done.

V. Revisions of Project Plan

We have no revisions to the project plan as we are progressing at a good tempo. We suspect we will be able to complete the original vision as described in the project specification.

[4]https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS