MALTE

Multi-access Live Text Editor

Progress Report 3

Adam Inersjö Jonas Norlinder Michael Rehn Joel Westerlund

December 12, 2019

1 Summary of Work

Last progress report we reported how a single-user editing experience had been achieved. This progress report has focused on a multi-user editing experience.

1.1 Features and Changes

General changes for frontend, backend and our shared utilities include code coverage reports so that we can get a general idea of where to focus on more testing. We also refactored the API documentation in the Wiki pages such that the endpoints use the same naming convention, consequently we made necessary refactoring in frontend and backend to reflect the new documentation.

Frontend

The most important addition to the frontend is a solution to the bug mentioned in the previous progress report: that caret positions was reset when another client sent an operation. With this solved, collaborative editing worked flawlessly. Together with this, we added caret synchronisation, so that you can see where another user are editing and what they are editing.

At the frontend we now have a login page displayed before you get to the actual UI. You login, just as before, with GitHub. If you are not invited to the project, the login screen will stop you from accessing the system. This is visible in Figure 1.

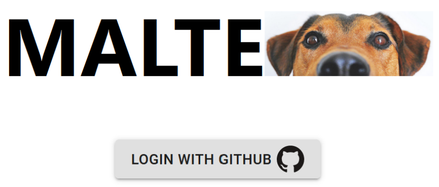


Figure 1: The login screen of the system.

We have also added a user list that shows each user's GitHub avatar in the top right corner. Next to this list, there is a button to invite new users to the system by typing their GitHub username into a text box. This will allow them to get pass the login screen mentioned above. We also have a welcome screen for when no file is currently open. This is all displayed in Figure 2.

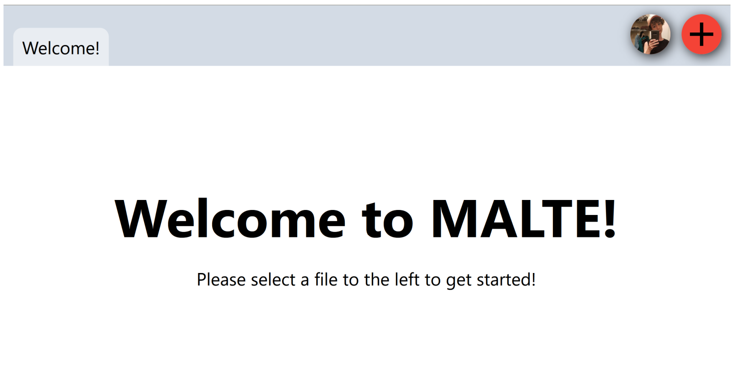
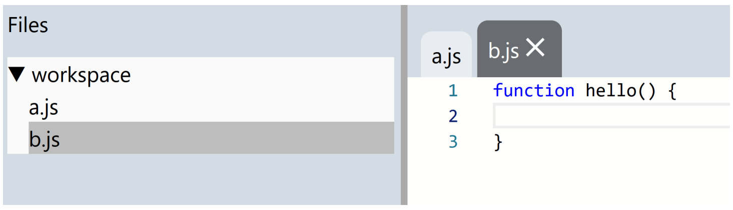


Figure 2: A welcome screen. In the top right there is the user list and the invite button.

Another feature we have added is to allow a user to open multiple tabs of files. The tabs provide easy access to the files you edit the most, so that you don't have to find them in the file tree each time. This is shown in Figure 3.



```
function hello() {

}
```

Figure 3: Tabs for easy access.

We have also re-worked the UI to make resizing of different areas more robust and added some tweaks to improve UI, e.g. changing the cursor while hovering something clickable in the file tree, or make the file tree auto-expand the root instead of being collapsed from the start. With this, we added a bottom status bar which displays the hash of the commit of which the build was built on. The bottom bar also contain a toggle for changing the editor theme from light to dark. This is shown in Figure 4.

Figure 4: Bottom bar with build information as well as dark mode.

Backend

The backend has been changed to accommodate for the changes discussed in frontend. For example, to be able to have a login page and user permissions, we needed a database to store user information. When starting a new instance of the server, no users would exist in the permission list, and thus, no one can invite any other users. To solve this problem, we have a "first time setup" mode, which will accept the first authenticated user irregardless if they are invited or not. This user can be seen as the owner of the project and can later invite other users.

We also needed a way to authenticate the user's WebSocket connection. Traditional ways of authentication is with a token, potentially stored as a cookie. However, WebSocket connections have no support of cookies and thus another solution had to be found. What we do is a two-step process.

The first step of authenticating the WebSocket connection is to authenticate over HTTP in the traditional sense described above, with a token stored as a cookie. When the client has received this token and gotten a confirmation that authentication was successful, it will send this token over the WebSocket connection. The backend server will then store this token along with the connections unique id. Thus, when this connection makes a request, we can lookup whether or not the token is stored with it to decide whether or not the connection is authenticated.

The backend has also received some well-needed refactoring to improve ease of testing and adding new functionality.

1.2 Changelog

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

2 Difficulties

We uncovered some nasty bugs regarding multiple editor tabs which were difficult to solve. The difficulty arose in the integration of the Monaco editor with React and our internal representation of editor tabs. After many attempts at fixing the bug, we realized that the simplest solution was to reorganize some code in a way that deviates from some of React's best practices.

3 Time Plan

According to our time plan, we are ahead of schedule. Our milestone this sprint was to allow for collaborative text editing which is now possible. Other tasks we had for this sprint was cursor synchronization and enabling user authorization to control access to the project server. These tasks are completed as well. As of now the rest of the sprint will be focused on finishing the draft report.

4 Next Sprint

There still exists some edge cases and bug fixes that needs to be addressed. Next sprint is solely dedicated to fixing known issues, testing the system to find eventual inconsistencies and writing the final report.

If there is time we also plan to improve the performance of the system. The way this can be done is by sending multiple operations to the RGA structure from the client in the one web socket message, hence reducing the overhead in communication.

5 Revisions of Project Plan

As we are ahead of schedule there is no need for any revisions to the project plan.