MALTE

Multiple Access Live Text Editor

Team Name: #dream-team

I. Members

Joel Westerlund
e-mail: jowe3751@student.uu.se
phonenr: 070 - 469 73 69

Jonas Norlinder (Initial Project Manager)
e-mail: jonas@norlinder.nu
phonenr: 076 - 145 90 39

Adam Inersjö
e-mail: adam.inersjo.2778@student.uu.se
phonenr: 070 - 732 04 60

Michael Rehn
e-mail: michael.rehn.2439@student.uu.se
phonenr: 073 - 91 588 52

We will change who is the project manager every sprint. The project manager role is partly administrative and partly focused on leadership. The project manager is responsible for

- keeping the website up-to-date
- making sure that project reports are written in time and handed in
- communicating with the teachers if any problems or questions arise
- ensuring that every member of the team is progressing on their respective tasks

II. Project Goals

The overarching project goal is to allow users to collaboratively edit and compile code. The system should have supporting functionality that makes this task easy, which could include access to terminal, commenting functions, chat functions, code high-lighting, etc.

III. Purpose and Function

The purpose of the project is to create an in-browser collaborative coding environment, in which one can edit and compile code on a server using a built-in shell. The environment runs in its own virtual machine to be able to provide a full development environment. The user will be able to use all commands as a normal Linux user. To provide the relevant software needed for development selected software such as compilers will be pre-installed on the server.

The system will be built upon a classical client-server architecture where multiple clients connect to a common server through a web interface. On the server side a project can be started and files edited by all clients simultaneously.

IV. Motivation

Editing locally on your own computer presents problems when pair-programming over distance. A collaborative code editing environment would solve this problem by allowing two, or more, people to see each other's changes in real time.

It is also very beneficial to have the code environment setup in the cloud, since this provides several advantages, namely: (1) ability to write code on any computer with a browser, (2) enabling use of a computer with limited hardware since all computing (e.g. compiling) is done at the virtual machine, (3) low-powered devices such as smartphones and tablets may be used.

Furthermore, if a user's code environment is set up in the cloud there is no need to configure your environment or replicate the code across multiple devices which saves the user time when switching between different devices e.g. between work, personal laptop and home desktop.

V. Related Work

There are already similar systems for collaborative, in-browser editing of code. These systems range from simple collaborative editors, such as Firepad[1], to full-fledged integrated development environments (IDEs) running on the cloud, such as Visual Studio Online[2]. Visual Studio Online can be accessed through a browser and extended to allow for collaborative editing in a project. REPL.IT[3] extends the concept of a project in Visual Studio Online by allowing for classrooms where students can learn and collaborate while teachers can assist the students and grade their solutions.

[1]https://firepad.io/
[2]https://code.visualstudio.com/docs/remote/vsonline
[3]https://repl.it/

Other related software that experiences synchronisation issues as ours are collaborative productivity tools such as Google Docs[4] and Figma[5], all of which either uses Operational Transformations[6] or Conflict-Free Replicated Datastructures[7][8] to handle synchronisation issues.

To improve performance of some classical CRDT structures, one can use an auxiliary structure as described in the paper High Responsiveness for Group Editing CRDTs[9]. The auxiliary data structure has no need to be synchronized between replicas, yet improves the upstream operation complexity considerably.

VI. Supporting Technologies

To host the server we will use a Linux Docker image to be able to easy start a new server at any computer or cloud provider, e.g. Amazon AWS or Microsoft Azure which can provide access to users independent of current location, with the only requirement of supporting the docker engine. The server backend will be implemented using TypeScript and Node.

A database on the server will keep track of user information and other relevant meta-data. Our intention is to implement the database using MongoDB.

The frontend will be implemented using TypeScript, HTML, CSS and React.

VII. Tools and Methodologies

During development of the project, Git will be used for version control with the repository stored on GitHub. GitHub Projects will be used as a project management tool to continuously plan and keep track of the project with a backlog of tasks. Group communication will be done using Slack.

A SCRUM-esque methodology will be used during the development of the project. The sprints will be two to three weeks long, each sprint beginning with sprint planning and ending with a retrospective held to improve the group's work and refine the backlog. The lengths of each sprint is presented below in the Time Plan. Daily "stand-up" meetings will be used to keep the group in the loop about daily progress.

[4]https://docs.google.com
[5]https://www.figma.com/
[6]https://en.wikipedia.org/wiki/Operational_transformation
[7]https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type
[8]https://www.figma.com/blog/how-figmas-multiplayer-technology-works/
[9]https://pages.lip6.fr/Marc.Shapiro/papers/rgasplit-group2016-11.pdf

VIII. Time Plan

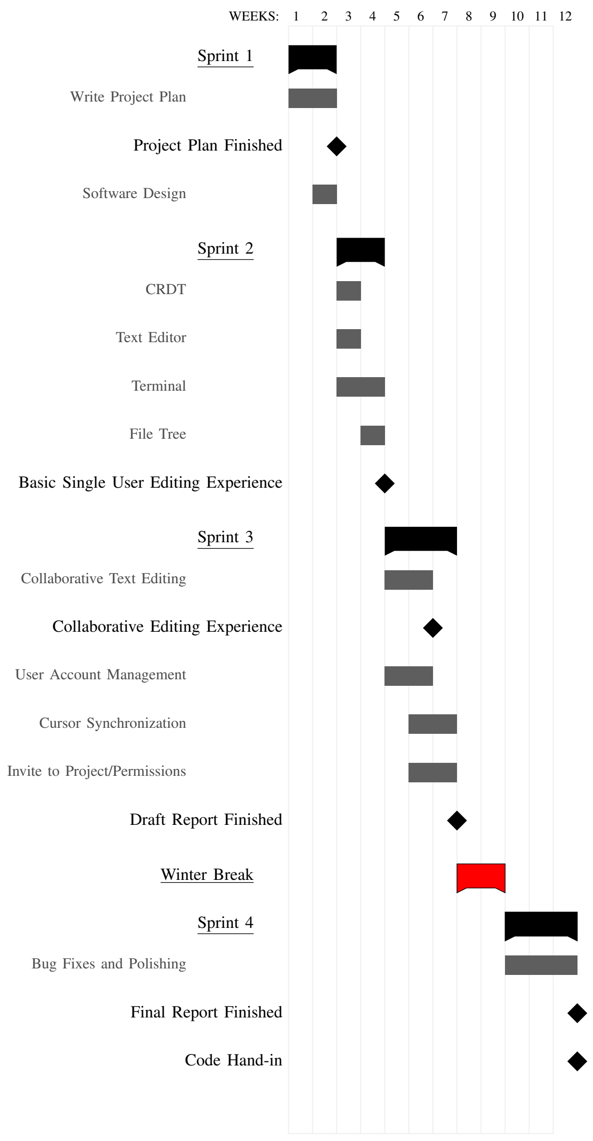

Fig. 1. Tentative Gantt planning