CodeLink: A Real-Time Collaborative Development Environment

Team Name: Stacked & Synced

Team Members

Dylan Everett (Project Manager)

Mark Burnette (Scrum Master)

Matthew Golden (Front End Developer)

Yan Tong (Back End Developer)

Advisor

Professor Christina Boucher

cboucher@cise.ufl.edu

[illegible]

GitHub Link

https://github.com/Dylante5/Stacked-and-Synced-CIS4914-Project

Abstract

Software teams and computer science classes often use too many separate tools for coding, testing, and sharing, which causes confusion and wasted time. Our project is a collaborative platform, like Replit or Live Share, that combines real-time code editing, version control, project planning, and chat in one place. This will make teamwork and teaching easier, faster, and more organized.

Table of Contents

Team Contract

Overall objectives: What the team is trying to achieve?

Modern development teams often rely on disjointed toolsets for creating, sharing, and testing code. Similarly, university professors frequently have to field questions from students using nonstandard development environments. Our goal is to create an integrated environment for real-time collaboration that reduces confusion between users and streamlines development.

Individual responsibilities: What each team member is expected to do?

Matthew Golden: Front end/UX developer

Mark Burnette: Front end/UX developer

Dylan Everett: Team/project manager, documentation

Yan Tong: Backend infrastructure developer

Values and Agreement Statements: Team agreements are written by everyone and they help define comfortable working parameters for everyone.

Our team will work towards the desired end of our project with efficiency, punctuality, and professionalism. Each member will make an effort to ensure they are not bearing too little or too much of the team's workload.

Software Configuration Management Protocol: refers to a set of defined procedures and standards used to track, control, and manage changes made to the software system to ensure consistency and quality of all modifications, management of merge conflicts, and version control.

Our team will be using GitHub to coordinate our project and keep track of changes made to the software. The build process and environment will be standardized and explicated for all team members.

Meeting times: When the team will meet outside the meeting with the advisor?

Our team will meet weekly Tuesday and Friday at 7:00 PM. For trivial discussions, or discussions that do not involve the team as a whole, issues can be brought up during a weekly meeting or conducted informally over Discord.

Meeting times with advisor: Frequency and Order of Meeting Facilitator (Everyone is required to lead/facilitate at least 1 meeting with the advisor; everyone must be present for all meetings with your advisor).

Biweekly meetings currently scheduled for Thursdays at 1:00 PM. Precise dates and times are subject to change. Facilitation will go in order as listed on this document above.

Communication: How team members will communicate with each other; what platforms?

Our team will meet and communicate primarily through Discord. Planning and organization will be done over a shared kanban board.

Conflict resolution protocol: How the team will solve problems?

Weekly sprints will be held, discussions on problems, working them out as a team rather than as a single person. Conflicts will be solved as soon as possible and with a neutral intermediary.

Consequences: What happens if someone violates the agreement?

The team will discuss any violations of the agreement, and if any individual does not take action going forward to correct the violation, we will escalate it to the instructor or teaching assistants of the course.

Signatures: All team members must sign the contract.

[illegible]

[illegible]

[illegible]

[illegible]

Introduction and Motivation

Modern development teams often rely on disjointed toolsets for creating, sharing, and testing code. Similarly, university professors frequently have to field questions from students using nonstandard development environments. We propose a full-stack collaborative development environment in the style of Replit or Visual Studio Live Share. Our goal is to create an integrated environment for real-time collaboration that reduces confusion between users and streamlines development. With applications both as a professional and pedagogical tool, users will be able to easily edit and share code in real-time. This will allow development teams to better coordinate build processes, share code, and collaboratively resolve bugs. For computer science educators, this will also allow for easier classroom collaboration and less time spent resolving students' errors.

Literature Review

The concept of an online, collaborative IDE is a relatively recent one with prototypes appearing around 2011, such as Collabode by Goldman et al. Since then, there have been multiple online, collaborative IDEs that have been released. These include Replit in 2018 and Code With Me by JetBrains in 2021. By looking at both these early and modern online IDEs, CodeLink can avoid issues that these IDEs might have had as well as establish what is needed for an online, collaborative IDE.

Arvue

Arvue was an IDE used for publishing small Vaadin, a tool for making Java web apps, applications. As Arvue's focus was limited, its architecture was relatively simple. Due to this simplicity, CodeLink can use this as a general starting point for its components. Some major components from this IDE that will be applicable to CodeLink are the code editor, resource manager, HTTP load balancer, and global controller. The code editor will be the main component of what the user sees, but the other components will be important for maintaining the load of users. The resource manager will keep track of memory usage to provide information for the global controller and to stop any misbehaving program. The global controller keeps track of scaling so that capacity of the server is not exceeded. Finally, the HTTP load balancer manages connections and configuration information about the programs that are running. (Aho et al.)

Collabode

Collabode was a "collaborative web-based IDE for Java" made for researching simultaneous coding on an IDE. To create this IDE, the researchers created an algorithm for simultaneous edits. The algorithm works by batching small edits together that are complete lines. While the IDE might show text from a collaborator to another, it does not integrate it until the line is finished. This is what CodeLink will aim for when it comes to multiple developers working on the same files. One issue brought up by users of this prototype was the inability to interact with collaborators' code when it was not integrated. Another was the lack of a changelog as the algorithm the researchers used batched edits together. Codelink will look to fix these problems and provide users with options to switch between the auto-integration present in Collabode or for manual integration. (Goldman et al.)

Visual Studio Live

While not hosted in a web browser, Visual Studio Live is an extension to the Visual Studio IDE for online collaboration on the same files. The researchers here used Visual Studio Live to understand what users are looking for in an online, collaborative IDE as well as issues faced in the IDE that might be able to be addressed. Some desired features include auto saving for interruptions, automatic Git, protection on local files, more sharing features, sharing server information, and user permission for debugging. Some challenges faced include lagging, connection issues, edit conflicts, tool and plugin incompatibility, data safety, and inconsistency across various machines. These desired features and issues gives the team behind CodeLink some focus on what features to implement and safeties to put into place. (Tan et al.)

Proposed Work

All minimal viable product (MVP) goals will be worked toward first. Our primary target audience is software developers, and our secondary target audience is computer science educators. As these groups have significant overlap, the development of any of the below features will benefit both groups equally.

<u>Features:</u>

Minimal Viable Product:

- Real-time code editor complete with syntax highlighting and support for multi-file projects
- Compiler/interpreter interface with support for several languages
- Project planning dashboard (e.g. Kanban board) with task and team management
- Lightweight chat and comment system for remote communication
- Integrated terminal window for testing of CLI-based programs
- Version tracking/integration with GitHub

Stretch Goals:

- Remote container management for secure testing of more intensive applications
- Advanced linting/error-checking
- Optional AI-based assistant
- Mobile support (view-only)

<u>Languages/Technology Stack:</u>

- HTML, CSS, JavaScript, Python, SQL
- Frontend: React, Tailwind, Monaco Editor
- Backend: Node.js
- Collaboration: WebSockets or CRDTs
- Infrastructure: Docker, GitHub API

Product Backlog

Features (from highest to lowest priority) and who is responsible:

Minimal Viable Product

1. Real-time code editor complete - Dylan Everett
2. Compiler/interpreter interface with support for several languages - Mark Burnette
3. Integrated terminal window for testing of CLI-based programs - Matthew Golden
4. Version tracking/integration with GitHub - Yan Tong
5. Syntax highlighting and support for multi-file projects - Yan Tong
6. Lightweight chat and comment system for remote communication - Matthew Golden
7. Project planning dashboard (e.g. Kanban board) with task and team management - Mark Burnette

Stretch Goals:

1. Remote container management for secure testing of more intensive applications - Dylan Everett
2. Advanced linting/error-checking - Yan Tong
3. Optional AI-based assistant - Matthew Golden
4. Mobile support (view-only) - Mark Burnette

Project Plan

We aim to have:

9/19 - 1st MVP feature completed

10/3 - MVP features 2, 3 and 4 completed

10/7 - Presentation 1

10/24 - MVP features 5, 6, and 7 completed

11/4 - Presentation 2

11/7 - Stretch goal 1 completed

11/16 - Stretch goals 2, 3, 4 completed

11/18 - Demo and documentation completed

11/20 & 11/21 - Senior Showcases

12/2 - Final Project

12/3 - Peer Evaluation

References

Aho, Timo, et al. “Designing IDE as a Service.” *University of Jyväskylä*, 2011, https://urn.fi/URN:NBN:fi:jyu-202504093155.

Goldman, Max, et al. “Real-time Collaborative Coding in a Web IDE.” *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology,* 2011, http://hdl.handle.net/1721.1/72493.

Tan, Xin, et al. “Understanding Real-Time Collaborative Programming: A Study of Visual Studio Live Share.” *ACM Transactions on Software Engineering and Methodology,* vol. 33, no. 4. doi: 10.1145/3643672.