W I D E

Web Integrated Development Environment

Group 17
Adviser - Dr. Goce Trajcevski

Michael Davis - Database Management and Meeting Facilitator
Lily Krohn - Front End Developer and Meeting Scribe
Chris Lopez - Backend Developer and Record Manager
Kyle Marek - Front End Developer and Testing Engineer
Zachary Mohling - Full Stack Engineer and Project Manager
Griffin Stout - Front End Developer and Web Manager

Team Email: sddec20-17@iastate.edu
Team Website: https://sddec20-17.sd.ece.iastate.edu

Executive Summary

Development Standards & Practices Used

- Agile Development
- Continuous Integration / Continuous Delivery (CI / CD)
- Test Driven Development (TDD)
- Potentially Use Pair Programming
- Source Control (Gitlab)
- Task Management Software(Trello or Kanban)
- Backend Documentation (API Docs/Method Stubs)
- Organized file structure
- High level design
- Web design standards

Summary of Requirements

- Application is accessible via Web Browser
- Multiple users are able to collaborate on the same file simultaneously
- Application supports the compilation of C documents.
- Application supports an online text editor
  - Syntax Highlighting
  - Error Detection
  - Auto-completion
- Integrated version control via Git
- Server will store user files
- Backend will support direct peer to peer interaction via websockets
- Automatic file saving
- Support file version control
- Support live edits for files
- User will login to use the application via database verification
- We will need to utilize features of Google Cloud Platform
- Will also be utilizing open source softwares and libraries

Applicable Courses from Iowa State University Curriculum

- COM S 309 - Long term project management, team project management
- COM S 327 - Advance programming, large project development
- S E 329 - Project management
- S E 185/ EE 285 - Introduction into programming
- COM S 311 - Algorithms and advanced data structures
- COM S 363 - Database management systems
- S E 319 - Concurrent programming

New Skills/Knowledge acquired that was not taught in courses

- React.js
- Typescript
- Monaco Text Editor
- Kubernetes
- Google Cloud Platform
- Docker
- PostgreSQL

Table of Contents

1. Introduction

1.1 Acknowledgement

Special thanks to Dr. Goce Trajcevski for his help with project planning and development.

1.2 Problem and Project Statement

In a world that is becoming more dependent on cloud storage and cloud computing, there is more of a focus on making software development less dependent on the personal machine. However, almost all software is developed on a local machine by one collaborator at a time. Developing on a machine requires additional steps to share files and download files from the internet. On top of these additional steps, there is maintenance to ensure that the project will be able to run on additional machines. All of these tedious steps and maintenance just simply to run the project can be very time consuming and reduces the efficiency of software development.

The solution we propose to solve these issues is WIDE, a web integrated development environment. WIDE is a web application, meaning it will be accessible through a browser by going to the URL. Through WIDE, users are able to save files, develop software, and run projects all on the cloud. Users will no longer need to download files or upload files from their machine to the cloud. Neither will they have to be concerned with the project running on their machine, because all code can be compiled and executed on the remote server. To top it all off, WIDE will enable the ability for users to collaborate on files together with real time updates. WIDE is the path to a more efficient, and less dependent future for software development.

1.3 Operational Environment

The operating environment of WIDE will be from a device's web browser and served from various public cloud services. The targeted frontend environment will be web browsers of consumer desktop and laptop operating systems.

1.4 Requirements

Functional Requirements

- Collaborative editing of source files
- Built-in Git UI for version control
- Authentication of users for joining and logging into projects
- Autosaving and source history
- Create/Import project
- Execution output and interaction
- Compile and execute artifacts

Non-Functional Requirements

- Fast response time
- Must support an adequate amount of users
- Sufficient storage space available

1.5 Intended Users and Uses

Our intended user is a company with software development teams that doesn’t want to spend lots of money or time setting up and installing software on everyone's seperate computer. The intended use is for when teams want to work on the same code at one time without having to install software or have multiple people work together on the same computer.

1.6 Assumptions and Limitations

Assumptions

- Basic security will be implemented (such as database protection)
- Terminal interface is the only way to run a program (No GUI will be shown to the user if they create one in their program)
- Our application will only include text in English

Limitations

- Remote terminal security will not be included at first because it is not necessary for proof of concept of the project and is quite complex to implement
- C will be the only available language at first due to the fact that C is well defined and small, making it a good first language to incorporate
- Advanced security implementations will be lower priority compared to our main features
- Autosave functionality will start as a timed event, given enough time we will implement an autosave on difference within the files
- We will need to utilize features of Google Cloud Platform - budget around $500

1.7 Expected End Product and Deliverables

- Service-oriented Backend System
    - A microservice-based system which supports the functional requirements. Each service will be containerized and orchestrated to simplify deployment and provide resilience.
- IDE Web Application / Editor
    - Our front end will feature a web based-text editor where users can simultaneously edit, debug, and execute code. Our delivery date for this product will be December 2020, at the end of the semester.
- Documentation of Configuration Files

- ○ With the delivery of our application we will also provide documentation of the configuration files which will be used to define the build processes in order to run our app.

2. Specifications and Analysis

2.1 Proposed Approach

We have implemented a basic text editor that we will use to expand on for the rest of our project. We have done this by following our functional and non-functional requirements as listed in section 1.4. The rest of our approach method will be improving our basic text editor and implementing the functional requirements of collaborative editing of source files, a built-in Git UI, the authentication of users when joining and logging into projects, autosaving and source history, and many more. We will also have the non-functional requirements of fast response time, allowing for support of adequate number of users, and having sufficient storage space available.

2.2 Design Analysis

So far we have set up our gitlab repository, done research on the technologies we will need to implement in our project, and created a basic implementation of the text editor that our project will be based upon. We have made several conceptual diagrams to organize the layout of all of the components that will make up our project. Through meetings with our faculty mentor, we have identified assumptions and loose ends present in our project and have developed several use cases and scenarios to support our project ideas. So far we have made good progress and we will continue by solidifying our use cases and progressing on development based on this.

2.3 Development Process

We will be following an Agile Development Process featuring 2 week sprints. This will allow us to continuously develop and produce code and will allow for easy code refinement. We will also be executing a Test Driven Development process in order to prioritize testing throughout the creation of our project.

2.4 Conceptual Sketch

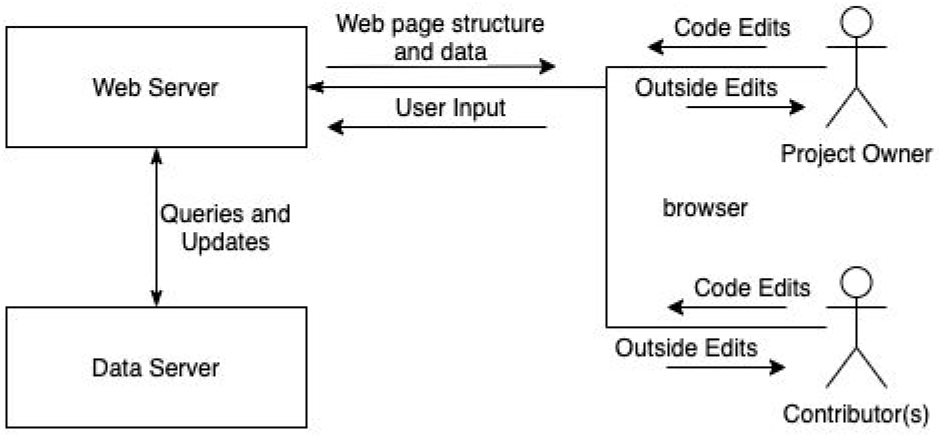

Figure 2.1

Multiple contributors will be able to make live concurrent edits in the same file through a web browser. The web browser interfaces with the web server to provide the content and data to the user and the data server. The projects will be hosted on the data server for all purposes. When the user opens a file, the data will be pulled from the data server and continually updated. If another user makes an edit, the changes will be sent to the data server and updated on all other user displays.

3. Statement of Work

3.1 Previous Work and Literature

There are a few applications that resemble similarities to our proposed project (ex: CoderPad), however they are lacking several things. CoderPad allows for simultaneous editing, however it lacks integrated version control. There are other well functioning IDEs, such as Visual Studio code, however there is no simultaneous editing. Our project will combine the features of simultaneous editing, auto saving, version control integration, and the functionality of a typical IDE to create a productive environment and allow for more efficient remote or in-person teamwork.

Sources used for research:

McKinnon, Jenni. "10 Best IDE Software for Web Development (2020)." *WebsiteSetup.org*, 1 Feb. 2020, websitesetup.org/best-ide-software/.

3.2 Technology Considerations

We've selected an ecosystem of technologies that have been proven in the industry to support efficient, reliable software systems. The following is a list of our chosen technologies, and their features which influenced our decision.

Infrastructure

Kubernetes

- Autonomous container orchestration
- Easy deployment
- Fault tolerance

Docker

- Provides a reproducible production environment
- Efficient alternative to virtual machines

Languages

Golang

- Standard library with built-in concurrency primitives
- Simple memory management
- Statically compiled into a single, versionable binary

TypeScript

- Strong typing system avoids increasing runtime error opportunities
- Included composite and utility types

Data

PostgreSQL

- Performant and scalable relational database
- Large community and corporate support and maintenance

Redis

- Fast key-value storage for server-side caching
- Simple and easy to use

Protocols

gRPC

- Fast and synchronous API calls
- Golang integration

WebRTC

- Supports peer-to-peer connections
- Best web protocol for real-time communication

3.3 Task Decomposition

The project can be broken down into some major tasks:

- Basic backend to frontend communication
- Account management and basic concurrent editing capabilities
- Concurrent editing implemented with much more functionality, version control integration defined, concurrent editing with version control, and terminal functionality allows the user to compile and run files
- Concurrent editing with version control is entirely complete and work is started on chat communication
- App is tested and all features are fully functioning

3.4 Possible Risks and Risk Management

Our only risks as of now are time constraints and technology decisions. As we get more into development, a risk we will have to consider is how we will handle security measures. We will be dealing with user information so we will need to take some security precautions, but it should not be the focus of our project. We will need to weigh these options and risks more at a further date.

3.5 Project Proposed Milestones and Evaluation Criteria

Some of our proposed project milestones include:

- A GUI the user can interact with
- A monaco-based text editor that allows simultaneous editing as well as autosave

- Comprehensive database system to handle both our internal and external file structures as well as users
- A way to share projects among users as well as register accounts and login
- Ability to import/create project as well as compile and execute it
- Integrate Git for version control
- A search feature that will allow users to look up projects based on keywords
- A chat feature that allows users to talk to each other

3.6 Project Tracking Procedures

We will be using GitLab's built in Scrum infrastructure to track our progress during this project.

3.7 Expected Results and Validation

Our end goal is to create a comprehensive development environment based on the web that allows teams to collaborate in real time, resulting in increased productivity without needing setup. A list of expected results is included in *Section 3.5*. We will verify that the results have been met by testing our own project within our browsers as well as allowing our advisor to test our project.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

1. **August 24th** - Work towards the project officially begins.
2. **September 1st -** Basic backend to frontend communication.
   a. **Frontend**
      i. The User Interface is reactive.
      ii. Frontend can make HTTP calls to the server.
   b. **Backend**
      i. The backend is able to perform CRUD operation with the database.
      ii. The backend is able to take HTTP calls from the frontend
   c. **Database**
      i. PostgreSQL database holds tables.
      ii. PostgreSQL tables are able to be modified.
3. **October 1st -** Account management is complete and basic concurrent editing capabilities.
   a. **Frontend**
      i. Accounts creation and login capabilities complete
      ii. Simple websocket connection with server and other users.
      iii. Able to view changes made by other remote users.
   b. **Backend**
      i. HTTP calls to manage account creation/editing are fully functional.
      ii. Able to connect to users via websockets.
      iii. Connect to Redis database is made.
   c. **Database**
      i. The PostgreSQL database is fully capable of managing user accounts.
      ii. Redis database is connected to the server.
4. **November 1st -** Concurrent editing is implemented with much more functionality. Version control integration is defined and we have begun implementing concurrent editing with version control. Terminal functionality is mostly complete and allows the user to compile and run files.
   a. **Frontend**
      i. The frontend allows the users to develop on the same file.
      ii. Project compilation and execution works correctly on the app.
      iii. First steps for version control integrating version control have been complete.
   b. **Backend**
      i. Server is able to store files on Redis database.
      ii. Server is able to compile and run projects and display output to the users.

        iii. Server manages concurrent editing through websocket connections.
    c. **Database**
        i. Redis database can store/update files.
5. **December 1st -** Concurrent editing with version control is entirely complete. Work is started on chat communication.
    a. **Frontend**
        i. The user interface is able to connect websockets specifically to any user that is on the same session in the editor.
    b. **Backend**
        i. The backend is able to manage version control capabilities such as branching, merging, and committing through libgit library.
    c. **Database**
        i. Redis is able to hold git objects.
        ii. Redis database functions completely as an internal git repository.
6. **December 11th -** App is tested and all features are fully functioning.

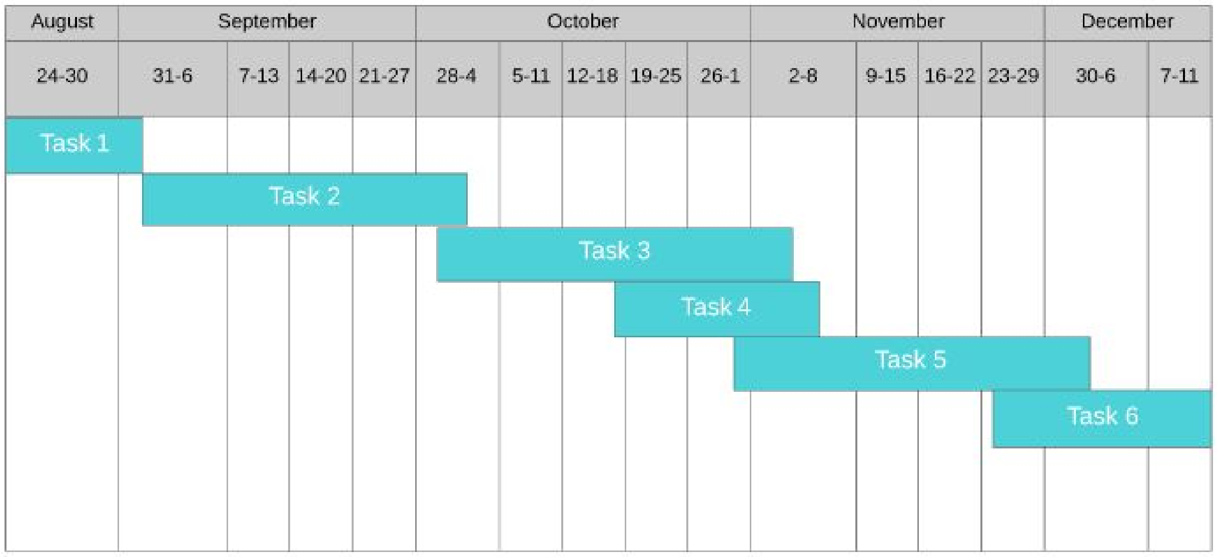

Figure 2.2

Task 1 - Basic backend to frontend communication.
Task 2 - Account login and creation.
Task 3 - Concurrent Editing Capabilities.
Task 4 - Terminal view and project compilation/execution capabilities.
Task 5 - Version Control Integration
Task 6 - Chat feature implemented allowing integrated collaborator communication.

4.2 Feasibility Assessment

Some foreseen challenges of our project are that we have lots of functionality that we hope to implement - however, due to the sake of time we will most likely have to cut back on some deliverables. We will aim to implement our main functionality first to our best ability and then expand from there.

4.3 Personnel Effort Requirements

Task Name	Date (s)	Effort Required
A GUI the user can interact with	Sept 1 - Sept 15	Mainly frontend members
A Monaco-based text editor	Sept 15 - Sept 30	Mainly frontend members
Comprehensive database system	Sept 1 - Oct 1	Mainly backend members
Import / Create / Compile project	Oct 2 - Oct 30	Both frontend and backend members, mainly backend members
Accounts / Login	TBD	TBD
Share Project Feature	TBD	TBD
Search Project Feature	TBD	TBD
Chat Feature	TBD	TBD

The table above displays a few of our tasks and their estimated start - end dates, along with the required effort in terms of which teams / members will need to focus on the task and possibly the estimated hours required to complete. We will aim to complete the first four tasks first, as these make up our basic product, and determine the rest based on time constraints.

4.4 Other Resource Requirements

Our project does not require any physical resources or materials as it is a software based project.

4.5 Financial Requirements

Our estimated budget for this project is $500. This will be used for public cloud infrastructure.

5. Testing and Implementation

5.1 Interface Specifications

5.2 Hardware and Software

5.3 Functional Testing

5.4 Non-Functional Testing

5.5 Process

5.6 Results

6. Closing Material

6.1 Conclusion

6.2 References

6.3 Appendices