WIDE

Team Members:
Michael Davis, Lily Krohn,
Chris Lopez, Kyle Marek,
Zachary Mohling, Griffin Stout
Advisor: Dr.Goce Trajcevski

Problem Statement

The modern software development process calls for a growing dependence on remote development and collaboration. However, development normally occurs on separate computers that all need to have the same system environments and access rights. This puts strain on a team that can't physically work together, and their systems that need to be increasingly more powerful and connected.

Solution

- Creating a collaborative setting where configuration and storage of projects are offloaded to the cloud.
- Offering joint, real-time editing of source files and GIT integration as well as project and team management,
- Allowing easy collaboration for work-from-home and geographically remote developers

Intended Users

- Any frontend JS developer or team which would prefer to keep project files on a consistent development environment
- Intended use is for when individuals or teams who want to work on the same code at one time
  - WIDE frees the user of having to install software or have multiple people working together on one computer.

Design Requirements

Functional Requirements

- Collaborative editing
- Built-in Git UI for version control
- Autosaving and source history
- Create/Import projects
- Authentication of users for login, projects, teams
- Execution output and interaction
- Compile and serve artifacts

Non-Functional Requirements

- Low text collaboration latency
- High scalability
- Quick bug and feature deployments
- High availability

Constraints / Environment

Constraints
- Time - main constraint
- Learning curves of new technologies
- Budget - $500

Environment
- Users can operate and run WIDE on their web browser of choice.

Design Approach

Frontend Components

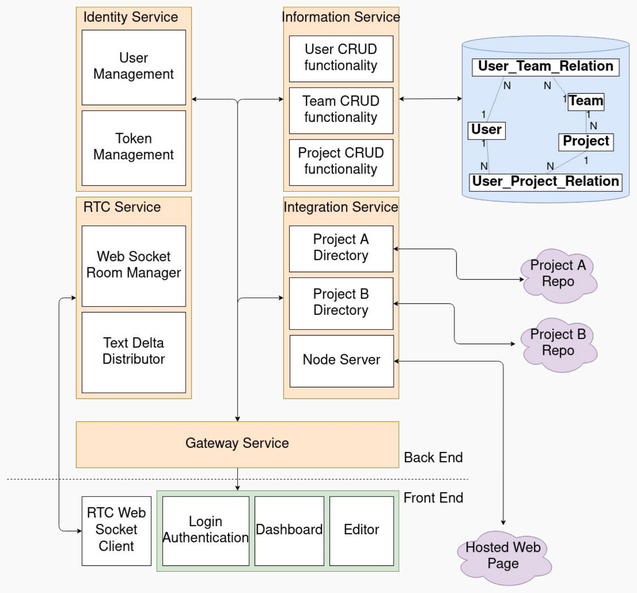
- **Authentication / Login**
  - Email validation, error handling
  - Confirmation email
  - Activation tokens and cookies
- **Projects & Teams Page**
  - Teams - create / add / remove users from teams
  - Projects - create / view projects
  - Project description
- **Monaco Editor & File Structure**
  - Autocomplete & syntax highlighting
  - Fetches files from server and displays file tree

Backend Components

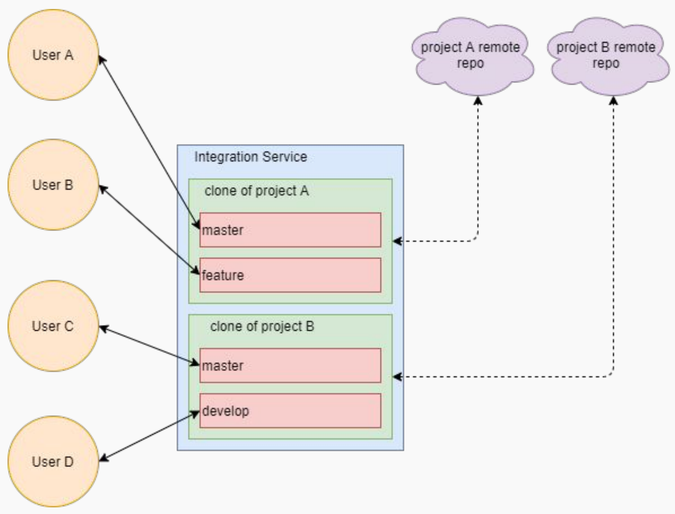
- **Identity Service**
  - User & Session management
  - User authentication
- **Information Service**
  - Interface with the DB
  - Provide dashboard entries
  - DB entity management
- **RTC Service**
  - Real-time communication
  - Cursor tracking, file saving
- **Integration Service**
  - Host project files
  - Provide project structure
  - Handle git- requests

System Diagrams

Block Diagram



Integration Service Diagram



Technical Details

Frontend

TypeScript / React
- Strong typing system avoids increasing runtime error opportunities
- Included composite and utility types
  Monaco Editor

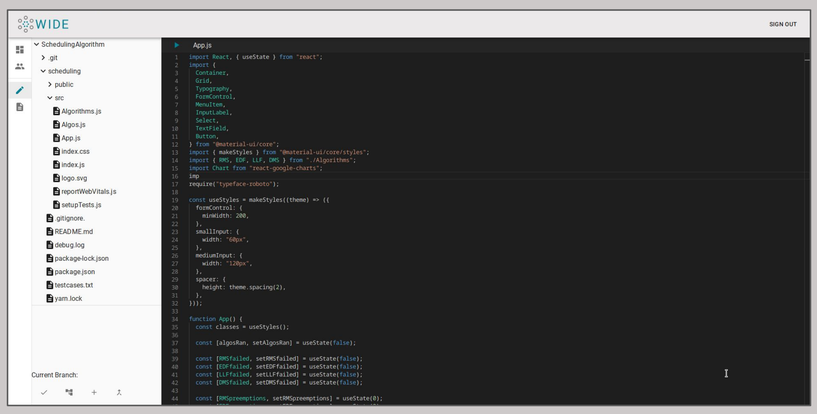
Backend

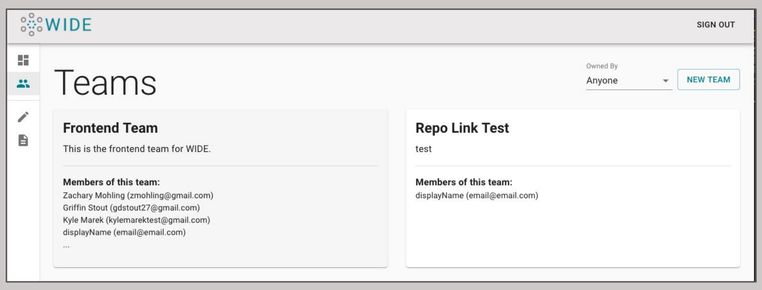
Kubernetes
- Autonomous container orchestration
- Easy deployment & fault tolerance

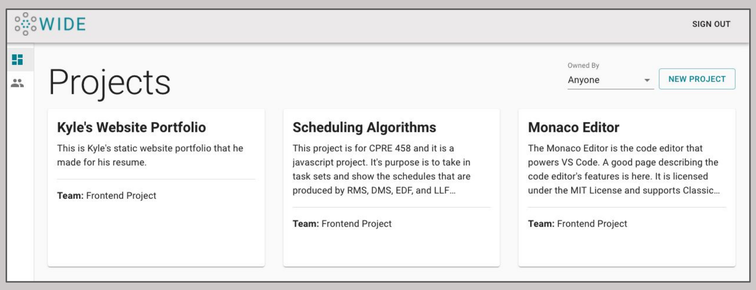
Docker
- Provides a reproducible production environment
- Efficient alternative to virtual machines

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

Final Product







Testing

Frontend Testing
- React-testing-library & Jest
- TDD

Backend Testing
- Postman
- Manual Testing of Microservices
- Built-in Go testing
- TDD