WebAssembly Based In-Browser Offline Code Execution

- Aditi
- Ashish Bhat
- Jayesh Pandey
- Mayank Agrawal
- Swati Lodha

Project Description

- The goal of our project is to support **offline (in-browser/no server) execution** of a random code snippet from languages like Go/ Python using web assembly.
- WebAssembly comes strongly handy because WebAssembly enables us to create a WebAssembly build of the compilers of Go/Python which can help **compile and execute the code** without the need of an internet connection.

Background:

- Currently, being able to learn programming online on platforms like Leetcode is **limited by having access to a stable internet connection.**
- The reason being that since **1996** , no language has been added to the list of languages for the web which allows code to run in the browser.
- JavaScript was the **only** full-fledged programming language that browsers understood, until the introduction of WebAssembly.

Project Goals:

- Develop a WebAssembly build for Go Compiler.
- Develop a WebAssembly build for Python Runtime.
- Develop a web application that has the following:
  - An online code editor supported by an open source code editor like Monaco.
  - Ability to run code and get execution results without an active internet connection.
  - Provide any input to the code snippet.
  - Stretch Deliverable - An interface where the goal is to solve a programming challenge with a set of predefined inputs.

Target Users

- Programmers
- Students
- Anyone who wants to learn programming

Main Functions and Unique Features

- Integration of webassembly build with monaco editor for syntax highlighting and to make ‘Run code’ work in the browser **without internet**.
- Ability to test the code with custom input (STDIN) and show the output of the code against the custom input.
- Ability to show the stack trace in case of errors.
- Database for storing programming problems on the platform.
- Ability to change the theme to light or dark based on user’s working environment and preferences.

Main Functions and Unique Features

- Ability to write code in multiple languages like Go/Python on the platform.
- Ability to see the STDOUT of the program so that the output can be read and errors can be debugged.
- Ability to stop the program at any given moment.
- Ability to save the code with its respective extension file on user's local machine.
- Ability to see the STDERR of my program so that errors like segmentation faults can be identified.

Technical Implementation:

These are the major technologies that we used to build this project:

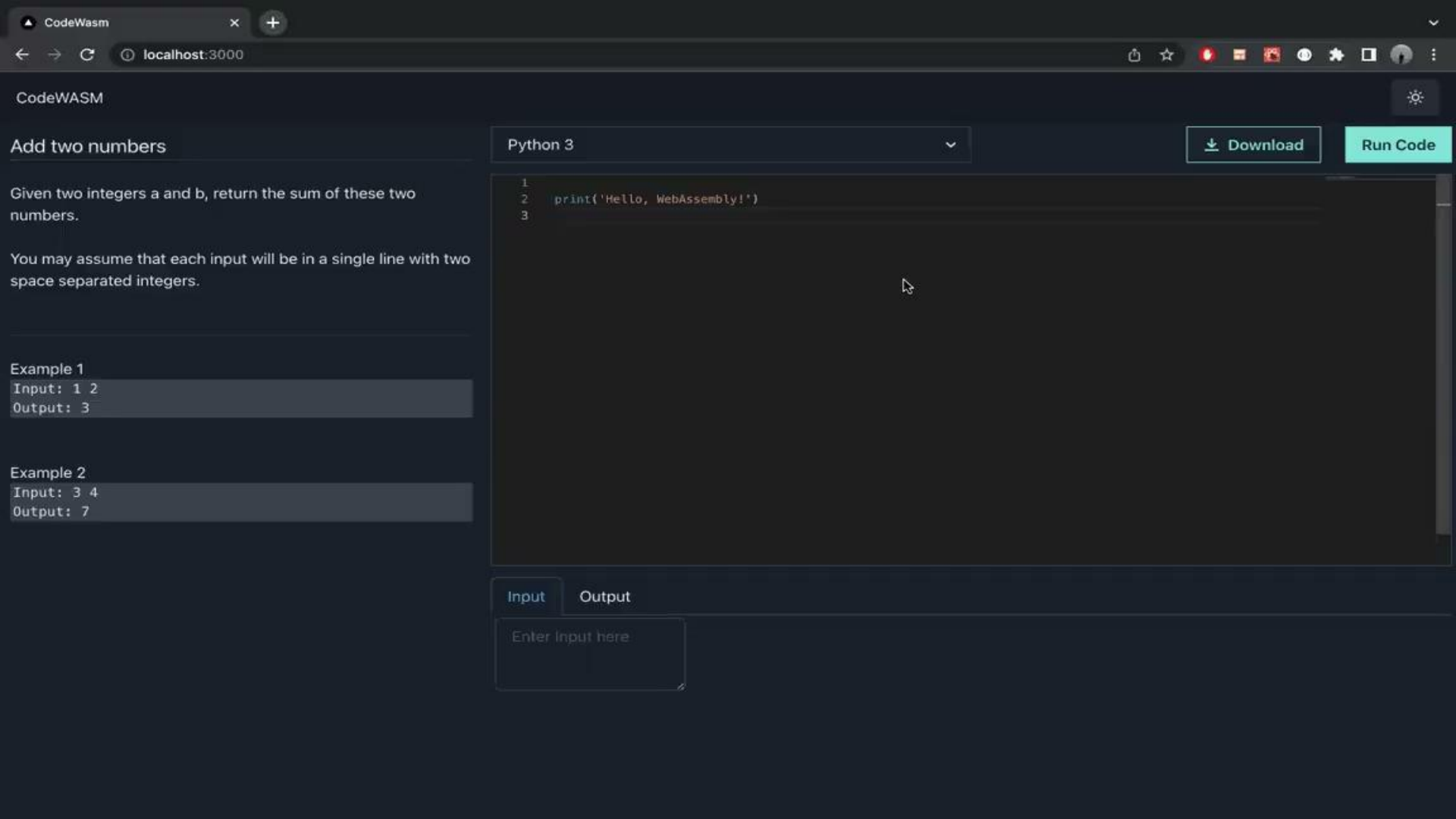
- WebAssembly
- ReactJS with NextJs
- The UI Library used for this project was called Chakra UI
- Monaco Editor
- MongoDB
- Git for collaboration

Difficulties and Challenges

- We faced difficulty with developing webassembly build for C++ , we acted quickly and instead focused on the next set of product features to get a working product out.
- We did not do a great job of estimating the time and effort it would take to implement some of the user stories while planning the backlog and that made us remove some of them in the middle of the sprint.

Lessons Learned:

- We need to dive in more deeper during our research phase
- Better planning is important to overcome obstacles and challenges faced during the software development life cycle
- The team learnt many lessons, both technical and managerial in terms of how to deal with tough situations.
- We had to make a tough decision to keep up with the project progress and we did well in that direction.


CodeWasm
localhost:3000
CodeWASM
Add two numbers
Given two integers a and b, return the sum of these two numbers.
You may assume that each input will be in a single line with two space separated integers.
Example 1
Input: 1 2
Output: 3
Example 2
Input: 3 4
Output: 7
Python 3
Download
Run Code
1
2 print('Hello, WebAssembly!')
3
Input
Output
Enter Input here

Thank You!