Online Code Execution Engine

MINOR PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

Pragyan
Enroll No: 03915602718

Pranav Gupta
Enroll No: 04015602718

Ritik Chawla
Enroll No: 05015602718

Sarthak Gupta
Enroll No: 05815602718

Under the Guidance
of
Ms Yamini
Principal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DR. AKHILESH DAS GUPTA INSTITUTE OF TECHNOLOGY & MANAGEMENT
(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)
NEW DELHI – 110053

DECEMBER 2021

CANDIDATES' DECLARATION

It is hereby certified that the work which is being presented in the B. Tech Minor Project Report entitled **" Online Code Execution Engine"**
in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Computer Science & Engineering** of **Dr. Akhilesh Das Gupta Institute of Technology & Management**, **New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during the period from **August 2021 to December 2021** under the guidance of **Ms Yamini , Principal.**

The matter presented in the B. Tech Minor Project Report has not been submitted by us for the award of any other degree of this or any other Institute.

Pragyan
Enroll No: 03915602718

Pranav Gupta
Enroll No: 04015602718

Ritik Chawla
Enroll No: 05015602718

Sarthak Gupta
Enroll No: 05815602718

This is to certify that the above statements made by the candidates are correct to the best of my knowledge. They are permitted to appear in the External Minor Project Examination.

(Ms Yamini)
Principal

Dr. Anupam Kumar Sharma
HOD, CSE

The B. Tech. Minor Project Viva-Voice Examination of **Pragyan, Pranav, Ritik Chawla, Sarthak Gupta** has been held on **…….**

(Signature of External Examiner)

ABSTRACT

The maturation of the web platform has given rise to sophisticated and disturbing net packages consisting of interactive three-D visualization, audio and video software program, and games.

With that, performance and protection of code on the net has emerge as extra important than ever. Yet JavaScript because the only builtin language of the web isn’t properly-geared up to meet those requirements, specially as a compilation goal. Engineers from the four primary browser vendors have risen to the assignment and collaboratively designed a portable low-level bytecode

known as WebAssembly. It gives compact representation, green validation and compilation, and safe low to no-overhead execution. In preference to committing to a selected programming model, WebAssembly is an abstraction over contemporary hardware, making it language-, hardware-, and platform-independent, with use cases beyond just the internet. WebAssembly has been designed with a proper semantics from the begin. We describe the inducement, design and formal semantics of WebAssembly and offer some initial enjoy with implementations.

ACKNOWLEDGEMENT

We express our deep gratitude to **Ms Yamini**, **Principal**, **Department of Computer Science & Engineering** for her valuable guidance and suggestions throughout our project work.

We would like to extend our sincere thanks to **Dr. Anupam Kumar Sharma**, **Head of the Department of CSE** for his time to time suggestions to complete my project work. We are also thankful to **Prof. (Dr.) Sanjay Kumar, Director, ADGITM, Delhi** for providing us the facilities to carry out my/our project work. We are thankful to **Ms. Pinky Yadav, Project Coordinator** for her valuable guidance.

Pragyan
Enroll No: 03915602718

Pranav Gupta
Enroll No: 04015602718

Ritik Chawla
Enroll No: 05015602718

Sarthak Gupta
Enroll No: 05815602718

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Introduction

The scope of this project is to allow developers to have rapid prototyping abilities in the convenience of their browsers.

Our team is working on building a code execution engine directly in the browser which can be used by developers to rapidly test and implement various prototypes without going through the hassle of setting up a new development environment on their local machines.

Using Web Assembly developers will be able to see instant results of their code without ever having to build or run.

Since the engine is built in the browser itself, HTML and CSS layout prototyping can be leveraged via iFrames, thus speeding up the development process.

This code execution engine will be beneficial in many ways-

1. Transpiling JavaScript code through the browser and providing instantaneous results.
2. Leveraging ESBuild and WebAssembly to achieve extremely fast code execution.
3. Implementing iframes for DOM manipulation.
4. Saving all data in JSON format so it can be shared with others.

1.2 Goal

The internet started out as a easy file exchange network however has now emerge as the most ubiquitous utility platform ever, accessible across a enormous array of operating structures and device sorts. through ancient accident, JavaScript is the simplest natively supported programming language on the web, its full-size utilization unmatched by means of other technologies to be had only via plugins like ActiveX, Java or Flash. due to JavaScript's ubiquity, rapid performance enhancements in cutting-edge Vms, and perhaps through sheer necessity, it has become a compilation goal for different languages. via Emscripten, even C and C++ packages can be compiled to a stylized low-stage subset of JavaScript called asm.js. yet JavaScript has inconsistent performance and a number of other pitfalls, specially as a compilation target. WebAssembly addresses the hassle of safe, rapid, portable low-degree code onthe internet preceding attempts at solving it, from ActiveX to local patron to asm.js, have fallen quick of homes that a low-level compilation goal should have:

●Secure, fast, and transportable semantics
●Safe to execute
●Speedy to execute
●Language-, hardware-, and platform-unbiased
●Simple interoperability with the web platform
●Secure and efficient representation:
● compact and clean to decode
● smooth to validate and collect
● clean to generate for manufacturers
● streamable and parallelizable

1.3 Need of the Application

Safe protection for cellular code is paramount at the internet, because code originates from untrusted sources. protection for cell code has traditionally been performed by means of supplying a managed language runtime such as the browser's JavaScript VM or a language plugin. managed languages implement memory safety, preventing applications from compromising person records or machine kingdom. but, managed language runtimes have historically no longer provided a good deal for transportable low-level code, which include memory-risky compiled C/C++ programs that do not need rubbish series but are inherently rapid.

Fast Low-level code like that emitted by a C/C++ compiler is generally optimized in advance-of-time. native system code, either written by using hand or as the output of an optimizing compiler, can make use of the full overall performance of a gadget. controlled runtimes and sandboxing techniques have typically imposed a steep performance overhead on low-degree code.

Portable The internet spans now not simplest many device instructions, but unique machine architectures, operating struct ures, and browsers. Code focused on the web have to be hardware- and platform-impartial to allow packages to run across all browser and hardware kinds withthe same conduct. previous answers for low-level code were tied to a unmarried architecture or have had other portability issues.

Compact Code this is transmitted over the community need to be as compact as possible to reduce load times, shop doubtlessly highly-priced bandwidth, and improve standard responsiveness. Code at the internet is generally transmitted as JavaScript source, which is some distance less compact than a binary layout, even if minified and compressed.

1.4 Scope

The initial model of WebAssembly supplied right here specializes in assisting low-stage code, specifically compiled from C/C++. A few crucial features are nonetheless missing for completely complete aid of this domain and could be delivered in destiny variations, which includes 0-cost exceptions, threads, and SIMD commands. Some of these capabilities are already being prototyped in implementations of WebAssembly. past that, we intend to conform WebAssembly similarly into an appealing target for excessive-stage languages through inclusive of applicable primitives like tail calls, stack switching, or coroutines. A rather vital goal is to provide get admission to to the superior and rather tuned garbage creditors which are constructed into all web browsers, accordingly getting rid of the principle shortcoming relative to JavaScript while compiling to the internet in the end, we count on that WebAssembly will find a huge range of use instances off the net, and anticipate that it will potentially grow extra feature essential to aid these.

CHAPTER 2: ENGINEERING

2.1 METHODOLOGY

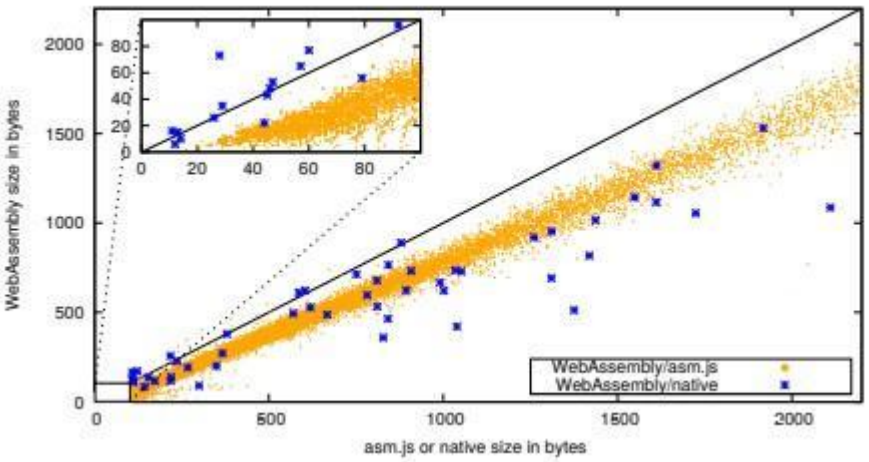
WebAssembly is the first answer for low-degree code at the internet that gives you on all the above layout desires. it's far the result of an unparalleled collaboration throughout primary browser vendors and an online community institution to construct a not unusual answer for excessive-performance packages. in this paper we focus on :

● A top level view of WebAssembly as a language that is the first genuinely move-browser solution for immediate low-degree code,

● An in-depth dialogue of its layout, together with insight into novel layout choices such as structured manipulate drift,

● A whole yet concise formal semantics of each execution and validation, such as a evidence of soundness, To our expertise, WebAssembly is the primary business energy language or VM that has been designed with a formal semantics from the start. This not only demonstrates the “actual global” feasibility of such an technique, however also that it ends in a significantly easy layout. at the same time as the internet is the primary motivation for WebAssembly, not anything in its design depends at the net or a JavaScript surroundings. it's miles an open well known mainly designed for embedding in multiple contexts, and we anticipate that stand-by myself implementations turns into to be had inside the destiny. The initial model usually makes a speciality

Fig 1.1 Binary size of



WebAssembly in comparison to asm.js and native code

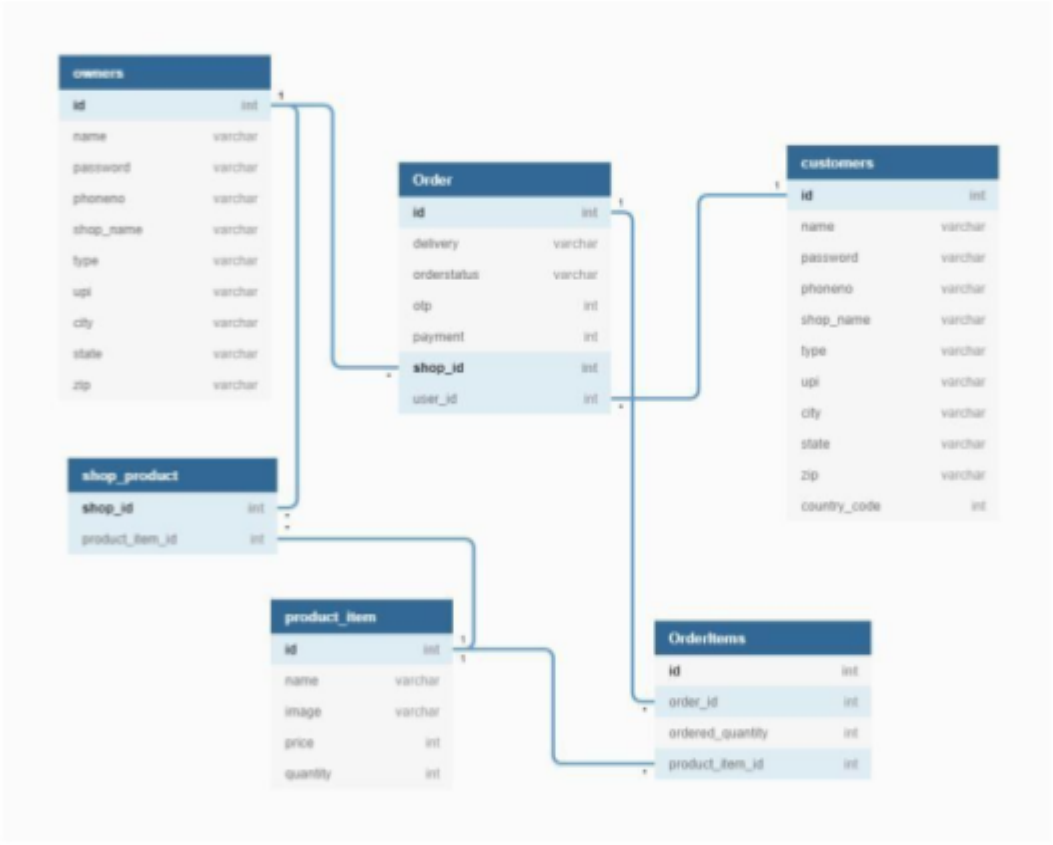
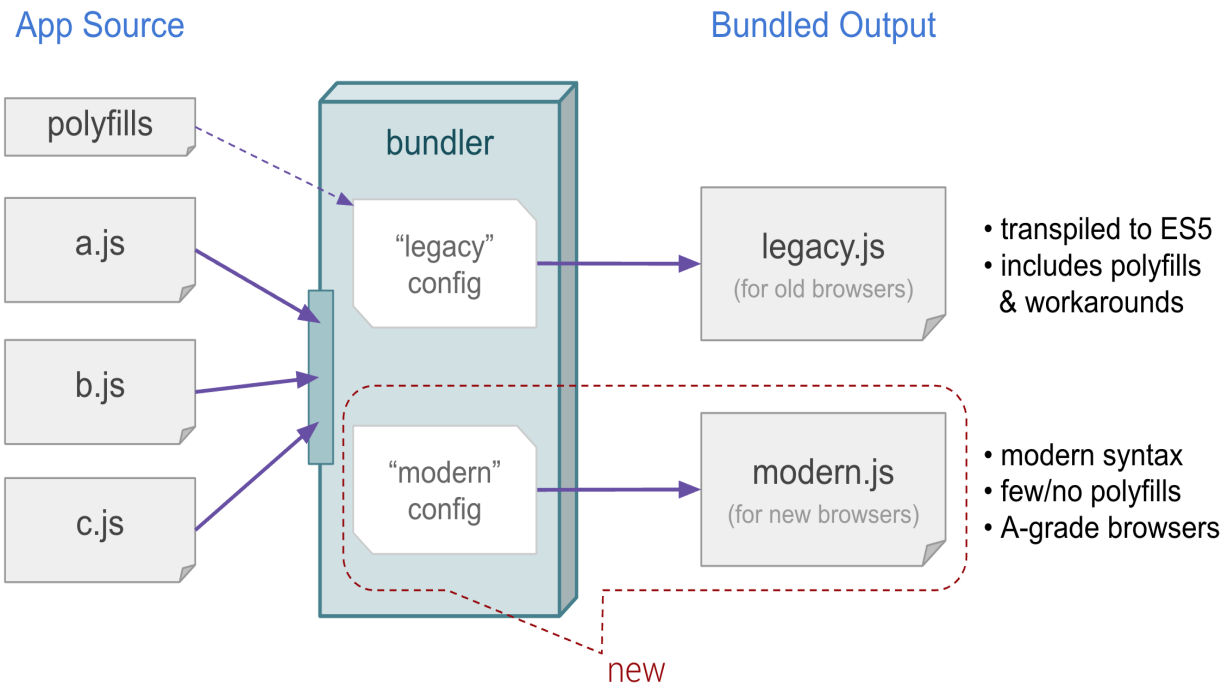


Figure 1.2 Database Design

2.2 Architecture of the Code Execution Engine



2.3 Interface

Technologies Used:

1. TypeScript – Programming Language
2. React – Frontend Framework
3. Redux – State Manager
4. EsBuild – A JavaScript bundler written in Golang
5. Bulma CSS – Styling library
6. iframes – for safe code execution

ESBUILD:

The main goal of the esbuild bundler project is to bring about a new era of build tool performance, and create an easy-to-use modern bundler along the way.

iFrames

An IFrame (Inline Frame) is an HTML document embedded inside another HTML document on a website.

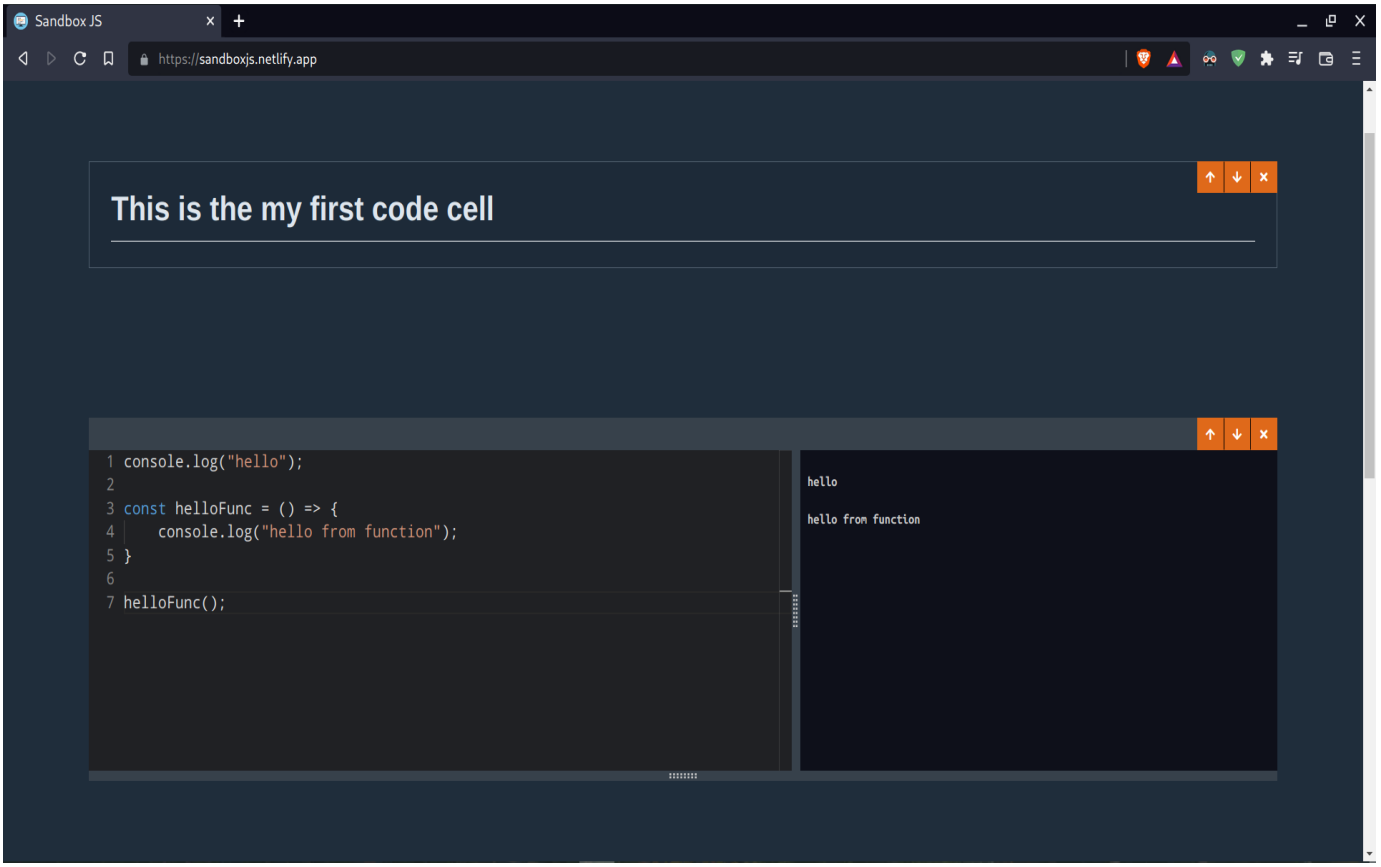
The IFrame HTML element is often used to insert content from another source, such as an advertisement, into a Web page.

Although an IFrame behaves like an inline image, it can be configured with its own scrollbar independent of the surrounding page's scrollbar..

CHAPTER 3: RESULT AND DISCUSSION

3.1 Result

https://sandboxjs.netlify.app/



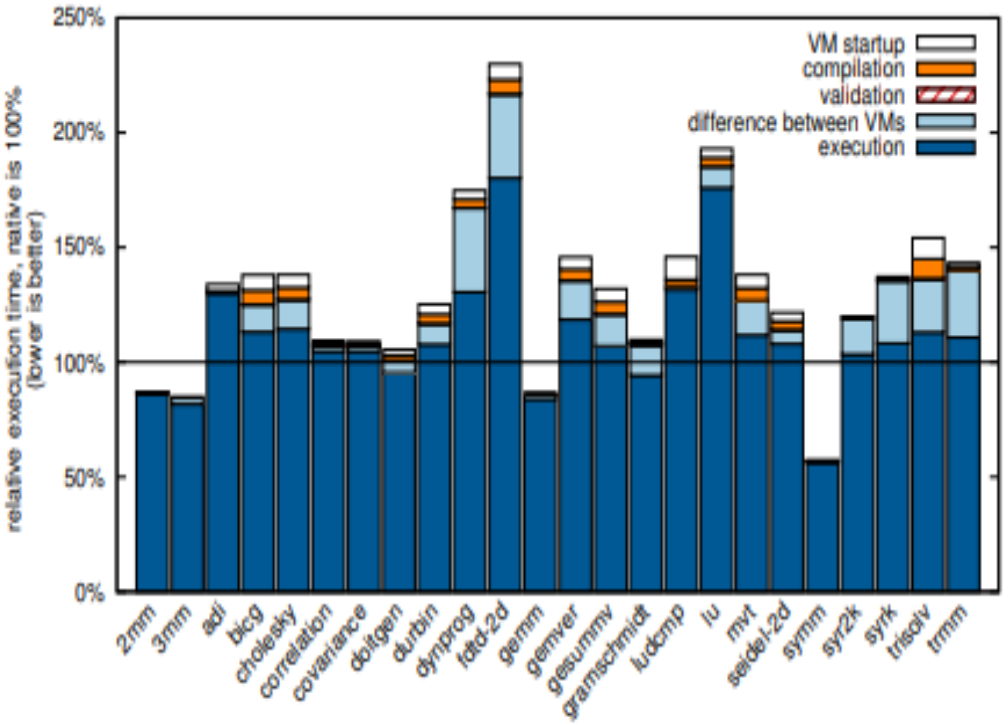


Figure 5. Relative execution time of the PolyBenchC benchmarks on WebAssembly normalized to native code

3.2 Future scope

The initial version of WebAssembly presented here focuses on supporting low-level code, specifically compiled from C/C++. A few important features are still missing for fully comprehensive support of this domain such as zero-cost exceptions, threads, and SIMD instructions. Some of these features are already being prototyped in implementations of WebAssembly.

1. ● Sandboxing for adding security layers

2. ● Interactive And Robust site execution

3.3 Limitations

A highly important goal is to provide access to the advanced and highly tuned garbage collectors that are built into all Web browsers, thus eliminating the main shortcoming relative to JavaScript when compiling to the Web. Finally, we anticipate that WebAssembly will find a wide range of use cases off the Web, and expect that it will potentially grow additional feature necessary to support thes

CHAPTER 4: CONCLUSION

This code execution engine provides solution for some real world problems. This application is capable to solve unnecessary resource setting time problem for libraries,plugins while doing development work .

It will encourage developers , testers to leverage their CPU resouces and prototype the projects.

REFERENCES

[1] Activex controls. https://msdn.microsoft.com/en-us/ library/aa751968(v=vs.85).aspx. Accessed:2016-11- 14.

[2] Adobe Shockwave Player. https://get.adobe.com/ shockwave/. Accessed: 2016-11-14

.

[3] ART and Dalvik. https://source.android.com/devices/ tech/dalvik/. Accessed: 2016-11-14.

[4] S. Nagarakatte, M. M. K. Martin, and S. Zdancewic. WatchdogLite: Hardware-accelerated compiler-based pointer checking. In Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization, CGO ’14, pages 175:175–175:184, New York, NY, USA, 2014.

.