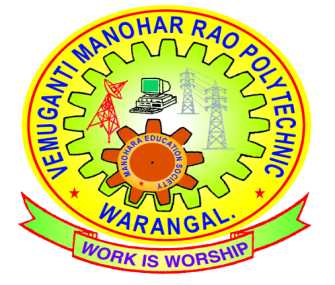
**A**

**PROJECT REPORT ON**

**CODE EDITOR**



SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF

**DIPLOMA IN COMPUTER ENGINEERING**

**SUBMITTED BY**

**A.SRIDATTA (20090-CM-015)**

**K.BHARGAVI (20090-CM-021)**

**V.VAISHNAVI (20090-CM-001)**

**V.DEEPIKA (20090-CM-014)**

**D.VIVEK (20090-CM-042)**

**MD.SHANNU (20090-CM-013)**

**D.VARSHITH (20090-CM-034)**

**P.ABHINAY (20090-CM-035)**

**K.AVINASH (20090-CM-058)**

**Under the valuable guidance of**

**Mr. B. VENKANNA**

**M. TECH**

**DEPT. OF COMPUTER ENGINEERING**

**VEMUGANTI MANOHAR RAO POLYTECHNIC**

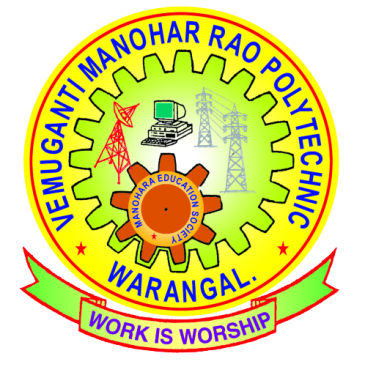
**Rampur, Warangal (TS)-506151**

**(2020-2023)**

**VEMUGANTI MANOHAR RAO POLYTECHNIC**

**(Sponsored by Manohara Educational Society)**

**Rampur, Warangal (T.S.)-506 151**



**CERTIFICATE**

This is to certify that this dissertation entitled

**“CODE EDITOR”** is carried out by

**A.SRIDATTA (20090-CM-015),** in partial fulfilment of the requirement for

**DIPLOMA IN COMPUTER ENGINEERING,** by the **State Board of Technical Education and Training (TS), Hyderabad,** is an award of bonafide work carried out by them under our guidance and supervision. The results embodied in this project report have not been submitted to any other institutions for the award of Diploma.

**PROJECT GUIDE HEAD OF DEPARTMENT**

**Mr. B.VENKANNA, M. Tech Mr. B.VENKANNA, M. Tech**

**EXTERNAL EXAMINER PRINCIPAL**

**Dr. P. Yellaiah**

**M. Tech, Ph. D**

**DECLARATION**

I declare that the work reported in the present entitled

**CODE EDITOR** is record of work done in the Department of Computer Engineering, VMR Polytechnic, affiliated to **SBTET,** Hyderabad and **AICTE**.

The result of this work has not been submitted to any university or any other institution for the award of any degree.

**SUBMITTED BY**

**A.SRIDATTA (20090-CM-015)**

**K.BHARGAVI (20090-CM-021)**

**V.VAISHNAVI (20090-CM-001)**

**V.DEEPIKA (20090-CM-014)**

**D.VIVEK (20090-CM-042)**

**MD.SHANNU (20090-CM-013)**

**D.VARSHITH (20090-CM-034)**

**P.ABHINAY (20090-CM-035)**

**K.AVINASH (20090-CM-058)**

**ACKNOWLEDGEMENT**

We wish to express our heartfelt gratitude to **Sri. Dr. P. Yellaiah,** Principal of

**VEMUGANTI MANOHAR RAO POLYTECHNIC,** Rampur, Warangal for providing all the facilities required for the seminar presentation.

We wish to take opportunity to express our sincere thanks to **Sri. B. Venkanna, H.O.D.** of Computer Engineering Department for the interest technical support and suggestions during the seminar leading to our success.

We are obliged and grateful to our guide **Mr. B.VENKANNA,** Lecturer in Computer Engineering Department for his valuable suggestion and sagacious guidance in all aspects during the technical of our training.

We hereby express our sincere thanks regards to all teaching staff and non-teaching staff, parents and friends who extended their helping hand in the accomplishment of the project.

**ABSTRACT**

**Introduction**

Code editors or source code editors are software that is designed specifically to help developers with coding. These are text editors with additional functionalities to manage and edit code. It can be standalone or it can be a part of an IDE. Using the best code editor can improve the speed of coding. Code editors are programming language-specific. Some editors support one or two programming languages whereas some support multiple programming languages. It can give suggestions and highlights based on language support.

Structure editor is a type of coding editor or we can say that it is the functionality that is included in the editors. Structure editing is used for manipulating the structure of a code based on the syntax tree. The syntax tree is nothing but the structure of the code which is written in a programming language. The code editors don’t compile the code. It just allows you to write and edit the source code.

**Existing system**

There are many possible causes for high memory usage. We've put together tools and a guide that can help you investigate potential performance issues. Start by using the process explorer. The process explorer shows the CPU and memory usage per child process of VS Code. Open it with the Open process explorer command. The process explorer should help you track down which processes are using the most memory.

Often times, an extension will turn out to be the root cause Also, even though you have only opened eight files, your workspace seems to be quite a bit larger than just those eight. Providing Intellisense and other advanced editor features often requires processing many unopened files as well. Whether or not the 2-3gb is justified or not is hard to say without understanding what extensions you are using and what your workspace contains. I recommend that you also take a look through the rest of our performance issue guide. It explains how to report performance issues and further investigate performance problems.

**Disadvantages**

1. Consumes a lot of memory to run code.

2. Unable to run freely on the mobile devices.

3. Unable to save in the local storages.

**Proposed system**

the proposed system is capable to use less memory to run the editor. It is platform independent. Ace is an embeddable code editor written in JavaScript. It matches the features and performance of native editors such as Sublime, Vim and TextMate. It can be easily embedded in any web page and JavaScript application. Ace is maintained as the primary editor for Cloud9 IDE and is the successor of the Mozilla Skywriter (Bespin) project.Ace is a community project. We actively encourage and support contributions! The Ace source code is hosted on GitHub and released under the BSD license ‐ very simple and friendly to all kinds of projects, whether open-source or not. Take charge of your editor and add your favorite language highlighting and keybindings.

**Advantages**

1. Consumes a less memory to run the code.

2. Able to run freely on the mobile devices.

3. Able to save in the local storages

**System configuration**

**Hardware configuration**

● RAM - 4gb

● ROM - 512gb

● Processor - Intel i3

**Software configuration**

● Language - javascript

● Libraries - PouchDB

● IDE - visual studio code

**INDEX**

Table of Contents

**1. INTRODUCTION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 01**

**2. EXISTING PROBLEMS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 03**

**3. EFFECTIVE SOLUTIONS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 04**

**4.** **ADVANCED TECHNOLIGIES FOR DEVELOPMENT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 05**

**5. FEATURES OF CODE EDITOR\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 07**

**6. SAMPLE SOURCE CODE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 09**

**7. OUTPUT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 25**

**8. PERFORMANCE ANALYSIS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 29**

**9.FUTURE SCOPE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 31**

**10. CONCLUSION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 32**

**11. BIBILOGRAPHY\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 33**

**1. INTRODUCTION**

A source-code editor is a text editor program designed specifically for editing source code of computer programs. It may be a standalone application or it may be built into an integrated development environment (IDE) or web browser. Source-code editors are a fundamental programming tool, as the fundamental job of programmers is to write and edit source code.

**objective**

the main objective of the project is to facilitate the code editing possible for anyone how is having access to a low end smartphone/laptop.

**Characteristics**

Source-code editors have characteristics specifically designed to simplify and speed up typing of source code, such as syntax highlighting, indentation, autocomplete and brace matching functionality. These editors also provide a convenient way to run a compiler, interpreter, debugger, or other program relevant for the software-development process. So, while many text editors like Notepad can be used to edit source code, if they don't enhance, automate or ease the editing of code, they are not source-code editors.

Structure editors are a different form of source-code editor, where instead of editing raw text, one manipulates the code's structure, generally the abstract syntax tree. In this case features such as syntax highlighting, validation, and code formatting are easily and efficiently implemented from the concrete syntax tree or abstract syntax tree, but editing is often more rigid than free-form text. Structure editors also require extensive support for each language, and thus are harder to extend to new languages than text editors, where basic support only requires supporting syntax highlighting or indentation. For this reason, strict structure editors are not popular for source code editing, though some IDEs provide similar functionality.

A source-code editor can check syntax while code is being entered and immediately warn of syntax problems. A few source-code editors compress source code, typically converting common keywords into single-byte tokens, removing unnecessary whitespace, and converting numbers to a binary form. Such tokenizing editors later uncompress the source code when viewing it, possibly prettyprinting it with consistent capitalization and spacing. A few source-code editors do both.The Language Server Protocol, first used in Microsoft's Visual Studio Code, allows for source code editors to implement an LSP client that can read syntax information about any language with a LSP server. This allows for source code editors to easily support more languages with syntax highlighting, refactoring, and reference finding. Many source code editors such as neovim and Brackets have added a built-in LSP client while other editors such as Emacs, vim, and Sublime Text have support for an LSP Client via a separate plug-in.

**Comparison with IDEs**

A source-code editor is one component of a Integrated Development Environment. In contrast to a standalone source-code editor, an IDE typically also includes debugger and build tools.Standalone source code editors are preferred over IDEs by some developers when they believe the IDEs are bloated with features they do not need

.

|  |  |
| --- | --- |
| **IDE** | **Code editor** |
| An IDE is a collection of software development tools that are intended to simplify the process of coding. It simplifies the entire process of developing software by combining the various components of a computer programme into a single GUI. | Text editors with sophisticated built-in capabilities and specific functionalities designed to ease and speed up the process of editing code are referred to as code editors. |
| Tools are Code Editors, Debuggers, Compiler and Interpreter | Tools are Simply code editor |
| Features are Text editing, compiling, debugging, syntax highlighting, unit testing, code completion etc | Feartures are Syntax highlighting, brace matching, auto completion |
| It consumes a lot of memory while running. | It requires less memory while running. |
| Some examples of IDEs are Eclipse, visual studio, Webstorm, etc. | Some examples of code editors are Atom, Sublime Text, Brackets, Visual Studio Code, Vim, UltraEdit, Bluefish, etc. |

There are many advantages as well as disadvantages of IDE or Integrated Development Environment. IDEs are not fundamentally different from code editors. Code Editor Plus. Many of the benefits offered to developers are obvious. They are the same as code editors and more. Unlike code editors, IDEs ensure that your code is bug-free and that you prepare and prepare (i.e., "compile") it so that it's ready to run when you're done. To do more. This saves time and intermediate steps and makes the process more efficient, whether you are a language-specific programmer. IDEs are advanced programs and may contain other features to facilitate programming.

**Controversy**

Many source-code editors and IDEs have been involved in ongoing user arguments, sometimes referred to jovially as "holy wars" by the programming community. Notable examples include vi vs. Emacs and Eclipse vs. NetBeans. These arguments have formed a significant part of internet culture and they often start whenever either editor is mentioned anywhere.

**2. EXISTING PROBLEMS**

Many of the students willing to develop projects on the mobile are facing issues in development of software with the code editors. They are trying to edit the code in the mobiles which has a smaller screen less computation power and bugs in mobile friendly code editors.

There are many editors in the market for editing the code. Where there are a lot of issues with them. One of the main problem is the memory issue many editors in the market use a memory of 1GB+ which makes a computer slow and effecting the development and productivity of the entire system. This problem is being highlighted. One of the report is

*“I've checked many similar issueson GitHub and StackOverflow, before I decided to submit mine. My issue is reproducible in both the stable and insider's version of VS Code. Extensions are disabled.*

*Within an hour or so VS Code starts consuming over 10GB of RAM. Closing it, frees up the memory immediately. If I miss the point, when the RAM consumption is too high, the OS hangs and I'm left with nothing but to hard reset the computer.”*

The another issue is many code editors in the mobiles a have issues with the user interface. The screen in the mobile is small while editing the code with auto code completion the size of seeing the code will be less than the actual size in the desktop. The icons, file headings in the editor will eventually scale down the size further more decreased. Some of the controls are only assigned with the mouse events. As a reason they won’t work when you use your touch screen.

While many of the editors on the web will also have to handle the problem of saving and storing the code on the local storage with memory issues. If effective measures are not taken the entire code of the programmer will be lost making an undesirable editor.

Finally, we have to deal with storage of the replicated code in the editor. Many of the code editors cannot support large code editing because of the read and write issues in the mobile. Mobile will not have enough read and write speeds to save the edited code automatically after some timely gaps.

**3. EFFECTIVE SOLUTIONS**

The problems identified in this project being solved using the following solutions. Firstly we need to solve the basic problem of the editors which is the large code handling quickly this can be done by one reading certain fragments of the file only when required to see. To perform the changes in the file we need to change only the certain fragments of the code by using the line number and the index position of the line in the file.

Second problem comes with the usage of the storage mode we cannot use the cookies as it cannot store MB,GB of the data. Where as the local storage has a limit of 5MB we cannot use this to store code with the size of more than 5MB. Even the indexedDB is having problems when using it in the safari browser which doesn’t support for older versions and only 50MB is allowed to store in the latest versions.

We are solving this storage issue with the pouchDB which allows to store upto 1GB of data in any of the browser without having any limitations. The stored code should not be lost by any of the cache clearing so we are ensuring that the cache storage is not cleared when the user clears the history. We are doing this by using the persist API.

Third solution is to solve the loading speeds of the editor which can be solved only if the editor is compressed or the editor should be lightweight. We made both the minification and the lightweight editor which solved most of the loading speed problems.

Fourth solution is the editors instant load time in the browser which make the editor to open and ready for editing the code in less than 1 second. We acquired such speed by optimizing the editor algorithm to load in fast way by changing the order of the algorithm into load UI of the editor then load the code modules finally load the editor extensions this made the main components to load first than load the remaining.

Fifth solution is the performance of the editor we are using the software performance management is the subset of tools and processes in IT Operations which deals with the collection, monitoring, and analysis of performance metrics. These metrics can indicate to IT staff whether a system component is up and running or that the component is behaving in an abnormal way that would impact its ability to function correctly much like how a doctor may measure pulse, respiration, and temperature to measure how the human body is "operating". This type of monitoring originated with computer network components, but has now expanded into monitoring other components such as servers and storage devices, as well as groups of components organized to deliver specific services and Business Service Management.

Sixth solution is the Progressive Web Applications often utilize what's know as an App Shell pattern. The App Shell is the bare minimum HTML, JavaScript and CSS that is need to show the static parts of the application. This usually includes things like the header, logo, navigation etc.

By caching the resources needed for the App Shell in a Service Worker, your application can load instantly on repeat visits and load at least the basic look and feel even if it's offline.

Finally we are using all the above methods to solve the problems of the code editor.

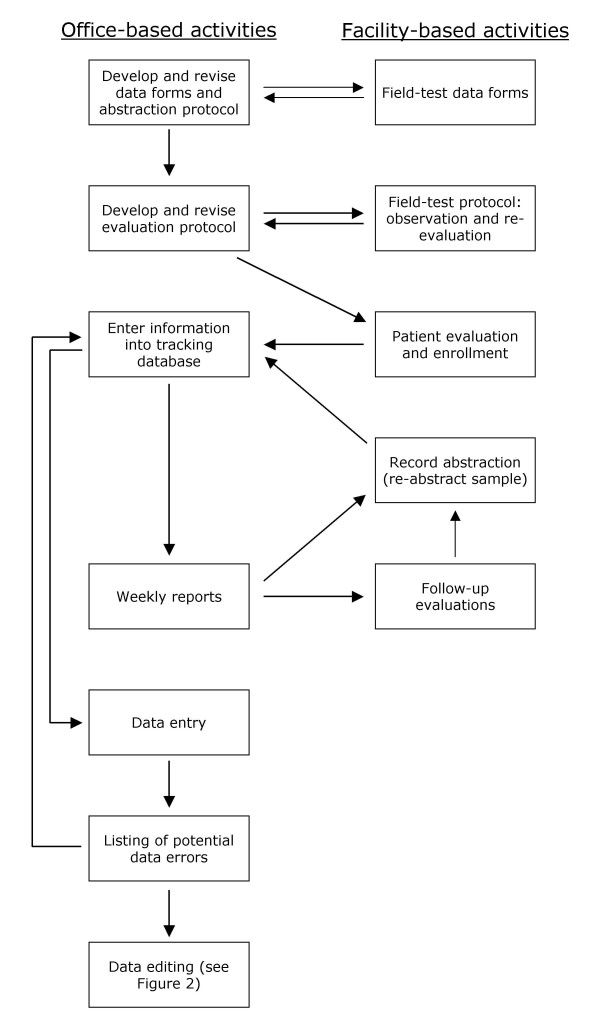
**4. ADVANCED TECHNOLIGIES FOR DEVELOPMENT`**

There are many technologies used to develop the code editor which are used to handle the installation of editor, offline support and code handling with cache. The technologies are

1. Progressive web application
2. PouchDB(for handling storage with cache)

**1. Progressive web application**

Platform-specific applications are known for being incredibly rich and reliable. They're ever-present, on home screens, docks, and taskbars. They work regardless of network connection. They launch in their own standalone experience. They can read and write files from the local file system, access hardware connected via USB, serial port, or bluetooth, and even interact with data stored on your device, such as contacts and calendar events. In these applications, you can do things such as take pictures, see playing songs listed on the home screen, or control song playback while in another app. Platform-specific applications feel like part of the device they run on. Progressive Web Apps (PWA) are built and enhanced with modern APIs to deliver enhanced capabilities, reliability, and installability while reaching anyone, anywhere, on any device with a single codebase.

****

**Capable**

The web is quite capable in its own right today. For example, you can build a hyper-local video chat app using WebRTC, geolocation, and push notifications. You can make that app installable and take its conversations virtual with WebGL and WebVR. With the introduction of WebAssembly, developers can tap into other ecosystems, such as C, C++, and Rust, and bring decades of work and capabilities to the web. Squoosh.app, for instance, uses this for its advanced image compression..

**Reliable**

A reliable Progressive Web App feels fast and dependable regardless of the network.

Speed is critical for getting users to use your experience. In fact, as page load times go from 1 second to ten seconds, the probability of a user bouncing increases by 123%. Performance doesn't stop after the onload event. Users should never wonder whether their interaction—for example, clicking a button—was registered or not. Scrolling and animation should feel smooth. Performance affects the entire user experience, from how they perceive your application to how it actually performs.

**Installable**

Installed Progressive Web Apps run in a standalone window instead of a browser tab. They're launchable from on the user's home screen, dock, taskbar, or shelf. It's possible to search for them on a device and jump between them with the app switcher, making them feel like part of the device they're installed on.

New capabilities open up after a web app is installed. Keyboard shortcuts, usually reserved when running in the browser, become available. Progressive Web Apps can register to accept content from other applications, or to be the default application to handle different types of files.

**2. PouchDB**

PouchDB is an in-browser database that allows applications to save data locally, so that users can enjoy all the features of an app even when they're offline. Plus, the data is synchronized between clients, so users can stay up-to-date wherever they go.

PouchDB also runs in Node.js and can be used as a direct interface to CouchDB-compatible servers. The API works the same in every environment, so you can spend less time worrying about browser differences, and more time writing clean, consistent code.

PouchDB supports all modern browsers, using IndexedDB under the hood and falling back to WebSQL where IndexedDB isn't supported. It is fully tested and supported in:

PouchDB also runs in Cordova/PhoneGap, NW.js, Electron, and Chrome apps. It is framework-agnostic, and you can use it with Angular, React, Ember, Backbone, or your framework of choice. There are many adapters, or you can just use PouchDB as-is.

PouchDB requires a modern ES5 environment, so if you need to support older browsers (IE <10, Android <4.0, Opera Mini), then you should include the es5-shim library. You can also use the LocalStorage and in-memory adapters, or fall back to a live CouchDB.

In Node.js, PouchDB uses LevelDB under the hood, and also supports many other backends via the LevelUP ecosystem. PouchDB can also run as its own CouchDB-compatible web server, using PouchDB Server.

**5. FEATURES OF CODE EDITOR**

The code editor has the features which make the editing much easier and faster. Our editor is having the following features

**Lightweight**

lightweight software also called lightweight program and lightweight application, is a computer program that is designed to have a small memory footprint (RAM usage) and low CPU usage, overall a low usage of system resources [citation needed]. To achieve this, the software should avoid software bloat and code bloat and try to find the best algorithm efficiency.

**Fast loading**

The entire editor loads in the rapid fast time from 0.2s - 0.8s which includes the time from transferring data through the network to the client device and loading the webpage in the browser in the device. Which make the website to load in less time and increase the user experience.

**Cross-platform**

As the editor is developed using HTML,CSS,JS we can run the entire editor in the web browser on any of the operating system which make this editor a cross-platform editor. Anyone who is having access to the internet and has a web browser can use this editor. Web editor is most useful than a native editor as it doesn’t require installation in your OS.

**Responsive design**

Responsive design is an approach to web design that aims to make web pages render well on a variety of devices and window or screen sizes from minimum to maximum display size to ensure usability and satisfaction. A responsive design adapts the web-page layout to the viewing environment by using techniques such as fluid proportion-based grids, flexible images, and CSS3 media queries, an extension of the *@media* rule.

**Progressive web application**

A progressive web application (PWA), or progressive web app, is a type of application software delivered through the web, built using common web technologies including HTML, CSS, JavaScript, and WebAssembly. It is intended to work on any platform with a standards-compliant browser, including desktop and mobile devices. Since a progressive web app is a type of webpage or website known as a web application, it does not require separate bundling or distribution. Developers can simply publish the web application online, ensure that it meets baseline installation requirements and that users will be able to add the application to their home screen.

Publishing the app to digital distribution systems like Apple App Store or Google Play is optional. As of 2021, PWA features are supported to varying degrees by Google Chrome, Apple Safari, Firefox for Android, and Microsoft Edge but not by Firefox for desktop.

**Auto code completion**

Auto code completion, which is similar to other autocompletion systems, is a convenient way to access descriptions of functions—and in particular their parameter lists. The feature speeds up software development by reducing keyboard input and the necessity for name memorization. It also allows for users to refer less frequently to external documentation, as interactive documentation on many symbols (i.e. variables and functions) in the active scope appears dynamically in the form of tooltips. Auto code completion uses an automatically generated in-memory database of classes, variable names, and other constructs that given computer code defines or references. The "classic" implementation of IntelliSense works by detecting marker characters such as periods (or other separator characters, depending on the language). When the user types one of these characters immediately after the name of an entity having one or more accessible members (such as contained variables or functions), IntelliSense suggests matches in a pop-up dialog. The user can either accept the suggestion by typing a statement-completion character (Tab or Enter) or a language-specific marker (such as the semicolon for C++), or continue typing the name. Over time, IntelliSense determines which variable or function the user most likely needs. IntelliSense also displays a short description of a function in the pop-up window—depending on the amount of documentation in the function's source code.

**Multiple languages support**

Syntax highlighting for over 110 languages. There are many languages in the market our editor supports them with the syntax highlighting and the new languages that are going to come can be added easily without any problem. We can also support these languages with the custom snippets of our own.

**Multiple themes support**

Themes let you to change the colours of the keywords, identifiers, constants, operators, comments even the background colour of the editor. This enables the programmer to choose his own choice of themes which could be light or dark for his convenience. This will also helps the programmers to reduce the strain on the eyes of the programmer.

**6.SOURCE CODE**

**index.html**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<meta http-equiv="content-type" content="text/html;charset=UTF-8" />

<link rel="icon" href="icon512.png" />

<title>Code Editor Download site</title>

<style type="text/css" media="all">

html {

background-color: #323232;

font-family: Sans-Serif, Arial, Helvetica;

font-weight: 350;

}

@media (min-width: 460px) {

body {

width: 460px;

display: block;

margin: auto;

}

img {

max-inline-size: 100%;

block-size: auto;

aspect-ratio: 2/1;

object-fit: contain;

}

}

img {

width: 100%;

height: auto;

}

.ki {

background-color: #323232;

-webkit-text-fill-color: transparent;

background: linear-gradient(#0892ff, #117cf5);

-webkit-background-clip: text;

background-clip: text;

}

@media (orientation: landscape) {

img {

height: 48vw;

width: 48vw;

display: block;

margin: 0 auto;

}

}

button {

text-align: center;

display: inline-block;

margin: 0 auto;

border: none;

background: linear-gradient(#0892ff, #117cf5);

color: white;

border-radius: 2pt;

padding: 6pt;

font-weight: 500;

font-size: 1.2em;

}

button:hover {

box-shadow: 0 0 0 2.5pt rgba(198, 226, 227, 1);

}

.p {

color: white; font-weight: 500; font-size: 1.3em;

}

.left {

text-align: left;

}

.cntr {

text-align: center;

}

.fnt {

font-size: 3em;

}

a {

text-decoration: none;

}

.trns-bgd {

background-color: rgba(255, 255, 255, 0.3);

padding: 1.2rem;

border-radius: 0.4rem;

margin-top: 20px;

}

.button-disabled {

opacity: 0.6;

cursor: not-allowed;

}

.button-disabled:hover {

box-shadow: none;

}

</style>

<link rel="manifest" href="manifest-editor.json" />

</head>

<body>

<img src="icon512.png" alt="" />

<div class="title">

<h1 class="ki cntr fnt">Code Editor</h1>

<div class="cntr">

<button type="submit" id="install">install</button>

<button onclick="openEditor()">open editor</button>

<div class="trns-bgd">

<h3 class="p left">Note: for firefox,safari the install button will not work

<br>

<br>please follow these steps to install

<br>1. click &#x22EE; options

<br>2.click install in your options

</h3>

</div>

</div>

<h3 class="p cntr">

The Code Editor allows you to write programs on tablet, mobile.

</h3>

<h1 class="ki fnt">Features</h1>

<dl>

<dt>

<h3 class="p left">Syntax highlighting for over 110 languages</h3>

</dt>

<dt>

<h3 class="p left">Over 20 themes</h3>

</dt>

<dt>

<h3 class="p left">Automatic indent and outdents</h3>

</dt>

<dt>

<h3 class="p left">Highlight matching parentheses</h3>

</dt>

<dt>

<h3 class="p left">Displays hidden characters</h3>

</dt>

<dt>

<h3 class="p left">Multiple cursors and selections</h3>

</dt>

<dt>

<h3 class="p left">Search and replace with regular expressions</h3>

</dt>

<dt>

<h3 class="p left">line wrapping</h3>

</dt>

<dt>

<h3 class="p left">Cut, copy, and paste functionality</h3>

</dt>

</dl>

<h1 class="ki fnt">Note</h1>

<h3 class="p left">please grant the presistant storage permission if asked by the browser.<br><br>you can code the

file which are lessthan 1200kb because of web storage issues.<br><br>please download the files regularly so you

code will be safe</h3>

<a href="license.html">

<h3 class="p cntr">Open Source License</h3>

</a>

<a href="editor.html" id="open-editor"></a>

</div>

<script>

navigator.serviceWorker.register('service.js');

const installApp = document.getElementById('install');

let installed = Boolean(localStorage.getItem('installed'));

if (installed) {

installApp.disabled = true;

installApp.innerText = 'installed';

installApp.setAttribute('class', 'button-disabled');

}

let deferredPrompt;

window.addEventListener('beforeinstallprompt', (e) => {

// Prevents the default mini-infobar or install dialog from appearing on mobile

e.preventDefault();

// Save the event because you'll need to trigger it later.

deferredPrompt = e;

// Show your customized install prompt for your PWA

// Your own UI doesn't have to be a single element, you

// can have buttons in different locations, or wait to prompt

// as part of a critical journey.

});

// Gather the data from your custom install UI event listener

installApp.addEventListener('click', async () => {

if (installed) {

installApp.disabled = true;

installApp.innerText = 'installed';

installApp.setAttribute('class', 'button-disabled');

return;

}

// deferredPrompt is a global variable we've been using in the sample to capture the `beforeinstallevent`

deferredPrompt.prompt();

// Find out whether the user confirmed the installation or not

const { outcome } = await deferredPrompt.userChoice;

// The deferredPrompt can only be used once.

deferredPrompt = null;

// Act on the user's choice

if (outcome === 'accepted') {

console.log('User accepted the install prompt.');

localStorage.setItem('installed', true);

installApp.disabled = true;

installApp.innerText = 'installed';

installApp.setAttribute('class', 'button-disabled');

} else if (outcome === 'dismissed') {

console.log('User dismissed the install prompt');

}

});

function openEditor() {

document.querySelector('#open-editor').click()

}

</script>

</body>

</html>

**license.html**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<link rel="icon" href="icon512.png" />

<title>Open Source License</title>

<style>

@media (min-width: 460px) {

body {

width: 460px;

display: block;

margin: auto;

}

}

body {

background-color: #323232;

font-family: Sans-Serif, Arial, Helvetica;

}

.ki {

background-color: #323232;

-webkit-text-fill-color: transparent;

background: linear-gradient(#0892ff, #117cf5);

-webkit-background-clip: text;

}

a {

text-decoration: none;

}

.p {

color: white;

font-weight: 500;

font-size: 1.3em;

}

.left {

text-align: left;

}

.cntr {

text-align: center;

}

.fnt {

font-size: 3em;

}

.sub {

color: #F2777A;

}

.title {

color: #66CCCC;

}

.trns-bgd {

background-color: rgba(255, 255, 255, 0.3);

padding: 1.2rem;

border-radius: 0.4rem;

}

</style>

</head>

<body><a href="index.html">

<h1 class="ki cntr fnt">Code Editor</h1>

</a>

<h1 class="sub cntr">Open Source License</h1>

<div class="trns-bgd">

<h1 class="cntr title">ACE editor</h1>

<h3 class="p left">Copyright (c) 2010, Ajax.org B.V.<br>All rights reserved.<br><br>Redistribution and use in

source and binary forms, with or without modification, are permitted provided that the following conditions

are met:<br><br> \* Redistributions of source code must retain the above copyright notice, this list of

conditions and the following disclaimer. <br><br> \* Redistributions in binary form must reproduce the above

copyright notice, this list of conditions and the following disclaimer in the documentation and/or other

materials provided with the distribution. <br><br> \* Neither the name of Ajax.org B.V. nor the names of its

contributors may be used to endorse or promote products derived from this software without specific prior

written permission.<br><br>THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND

ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY

AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL AJAX.ORG B.V. BE LIABLE FOR ANY

DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING

NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

POSSIBILITY OF SUCH DAMAGE.</h3>

</div>

</body>

</html>

**service.js**

self.addEventListener('install', (e) => {

e.waitUntil(caches.open('SDcodeeditor').then((cache)=>{

cache.addAll(

[ './editor.html',

'./ace.js',

'./ext-settings\_menu.js',

'./pouchdb-7.3.0.min.js',

'./eruda.min.js'])),);

});

// used to update the cache when service worker changes

self.addEventListener('activate', (e) => {

console.log('activated new service-worker');

});

self.addEventListener('fetch', event => {

if (event.request.url.startsWith(self.location.origin)) {

event.respondWith(

caches.match(event.request).then(cachedResponse => {

if (cachedResponse) {

return cachedResponse;

}

return caches.open('SDcodeeditor').then(cache => {

return fetch(event.request).then(response => {

return cache.put(event.request, response.clone()).then(() => {

return response;

});

});

});

})

);

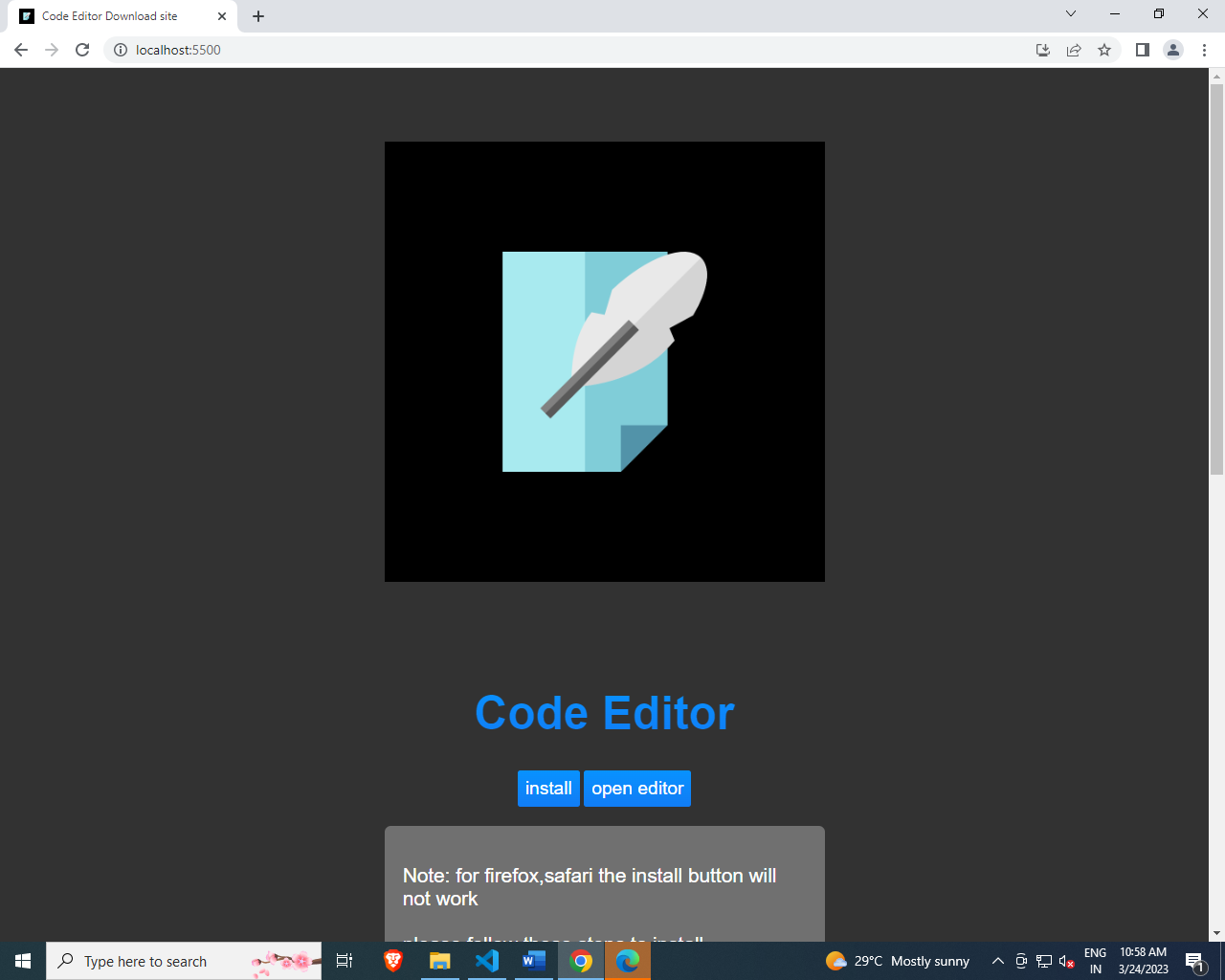
}

});

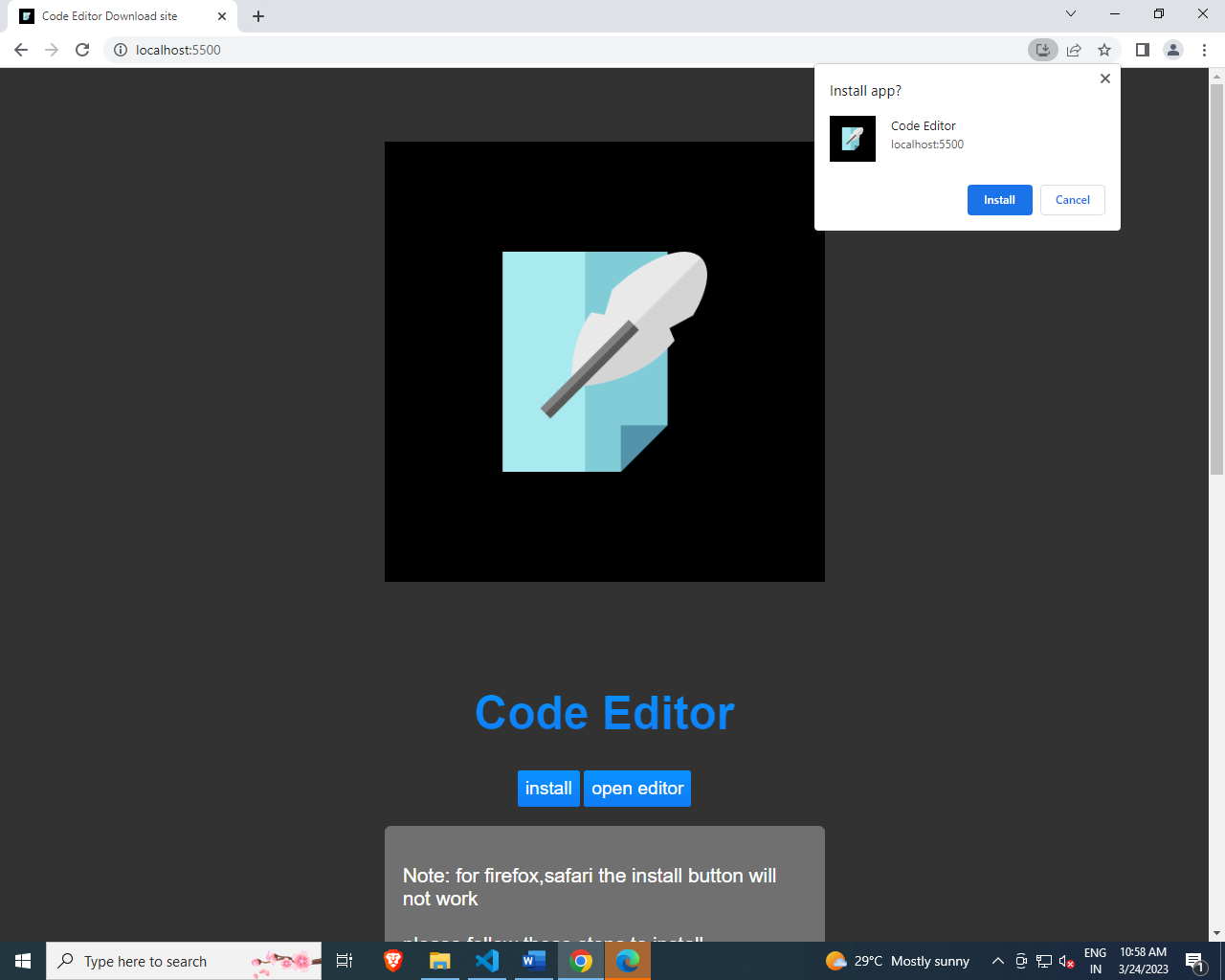
**7. OUTPUT**

**Sample outputs from desktop**

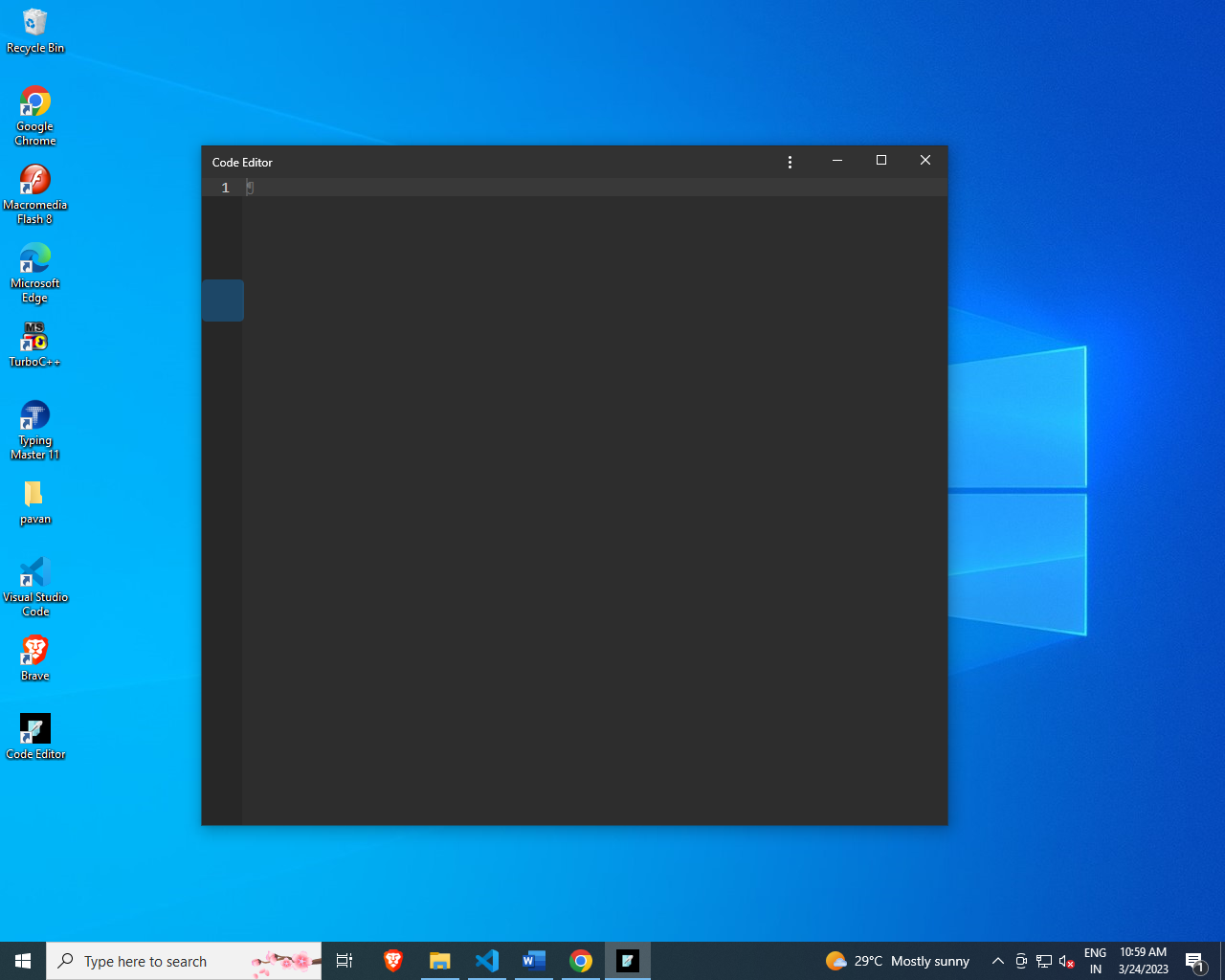
**Website home page**

****

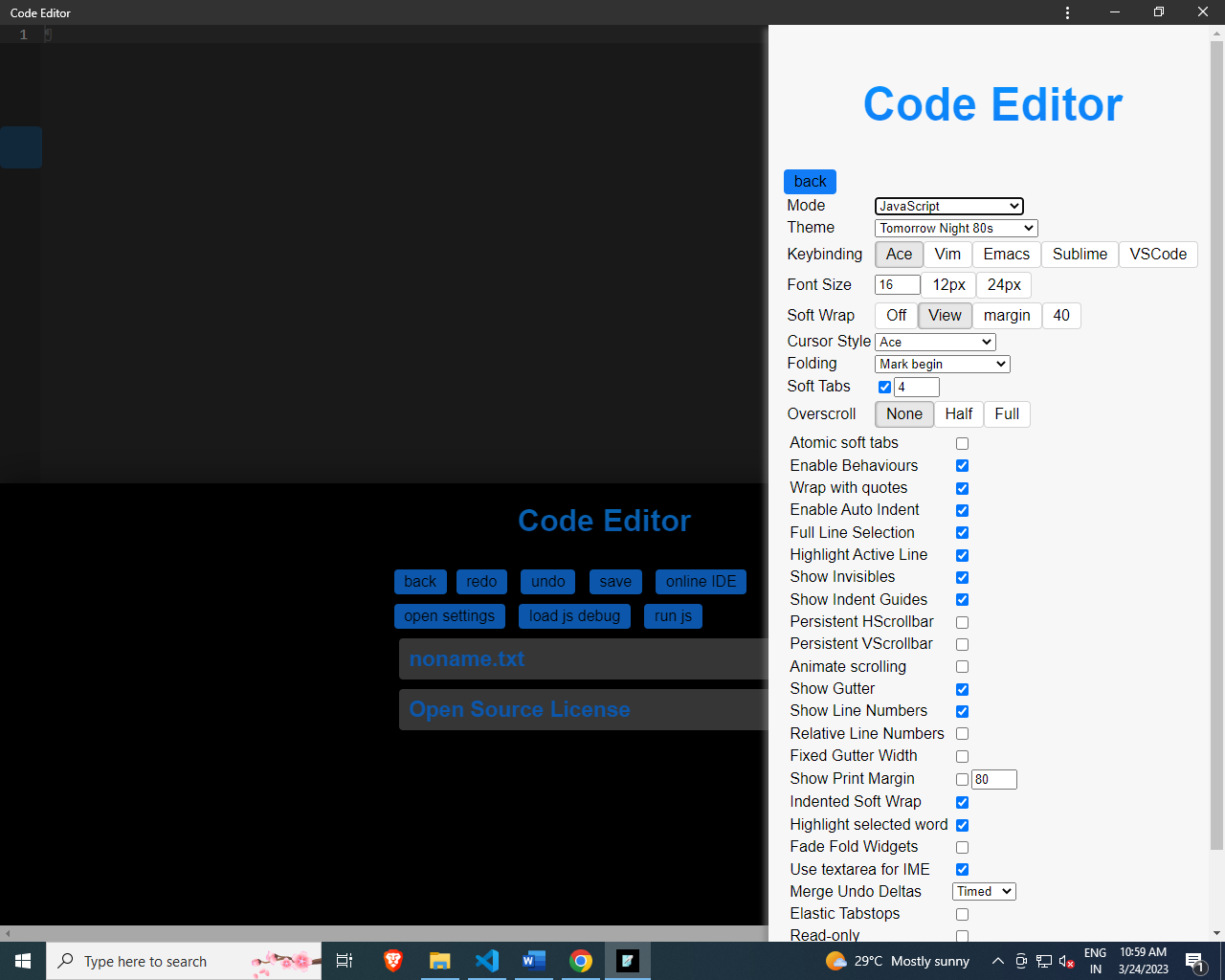
**Installing the editor**

****

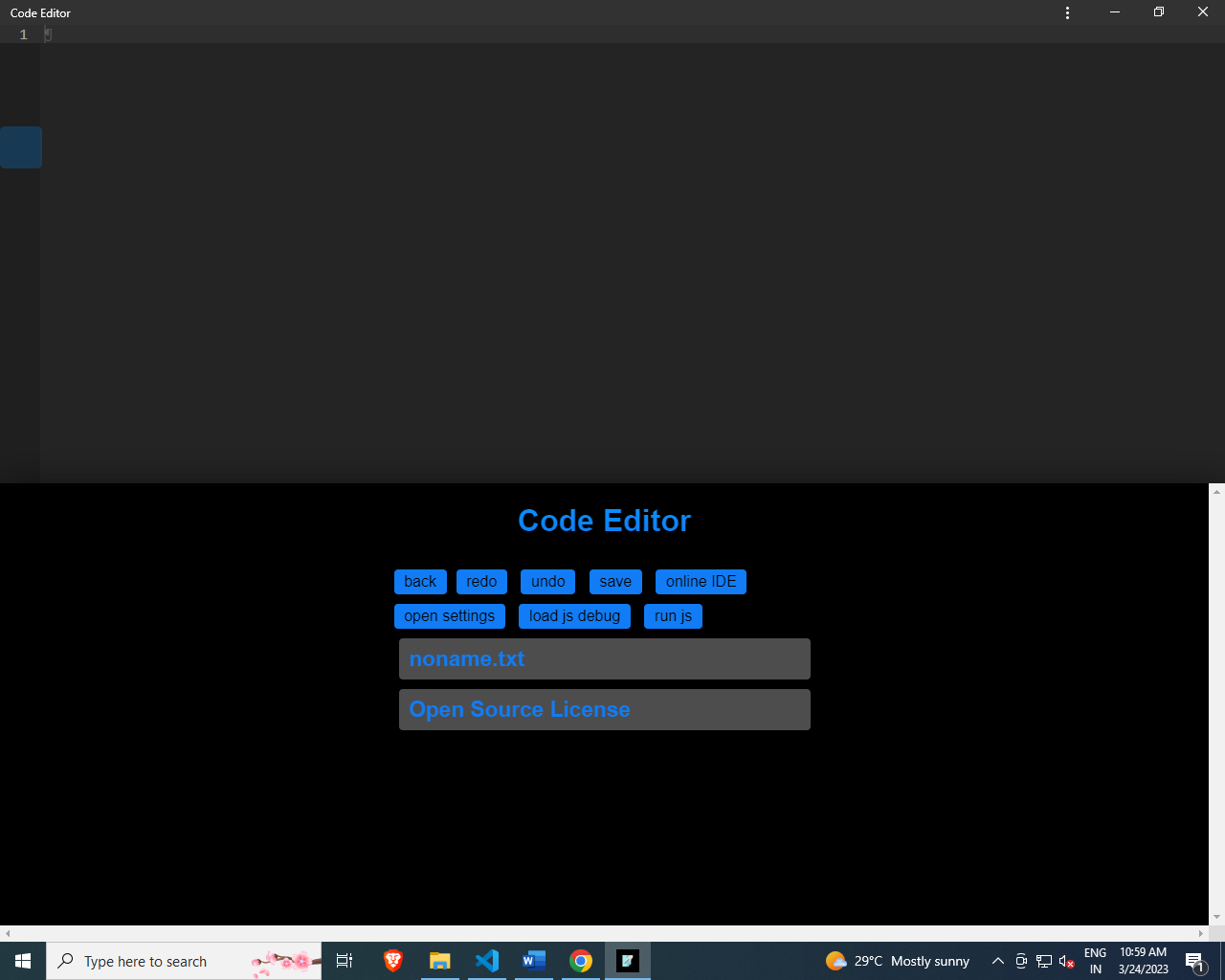
**After installing editor**

****

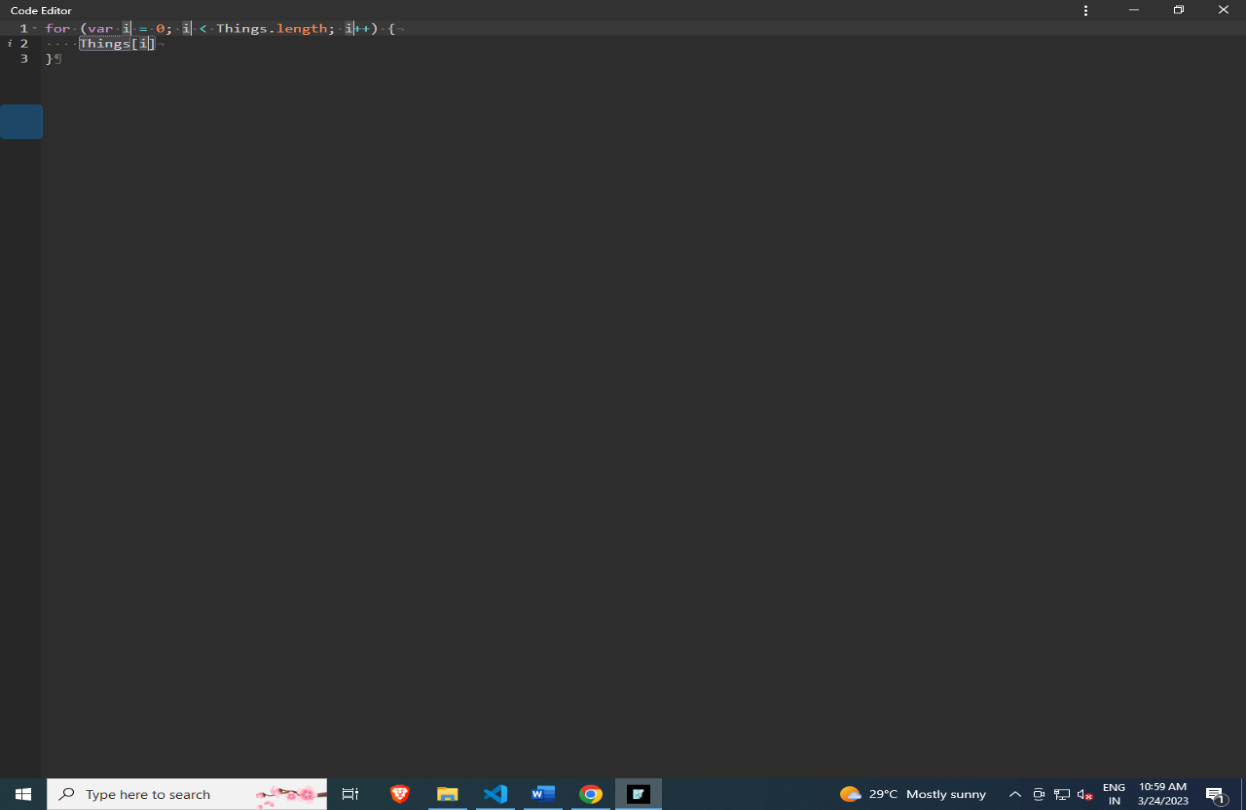
**Editor advanced settings**

****

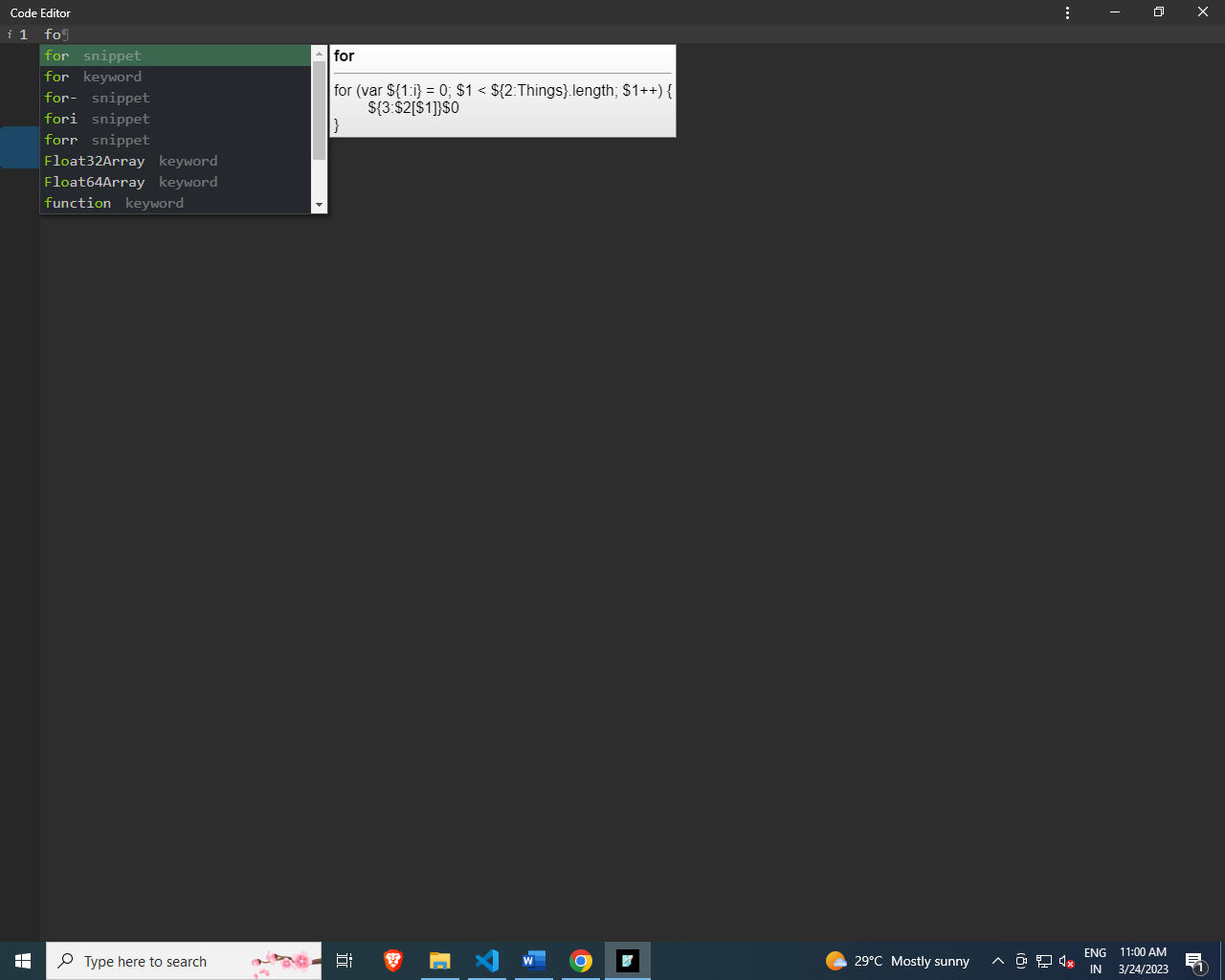
**Editor basic settings**

****

**Creating a javascript file**

****

**Auto code-completion**

****

**8. Performance Analysis**

The act of optimizing a software application to run as fast as possible on a given computing platform has never been a trivial task. In the past, developers achieved such goals by poring over hardware and software manuals, trying to locate the proper combination of assembly language instructions that would result in the level of performance that they desired. Since the computers they had available were all well documented and functioned in a wholly deterministic manner, it was relatively easy for a developer to determine the types of source-code adjustments that would work best on a given architecture.

In the past ten years, however, the software development landscape has evolved dramatically as the general public has embraced computing devices of all types and become increasingly reliant on them to accomplish everyday tasks. As the demand for more sophisticated applications has increased, designers have turned to use the use of higher-level languages and frameworks in an attempt to reduce development costs and remain competitive in the marketplace. As a result, applications have grown increasingly complex in terms of both code size and the interactions that occur within them. Therefore, while this "layered" approach to development may save time and money in the short run, it complicates the task of determining whether an observed performance issue is internal to an application or caused by the frameworks that it is built upon.  
  
Beyond this, computer hardware itself has been forced to change dramatically to keep up with the unrelenting demand for more computing power. The simple single-issue processors of the past have given way to super-scalar designs capable of executing multiple instructions in a single cycle while simultaneously reordering operations to maximize overall performance. As a result, the instructions passed into the processor have become merely a guideline for execution, as opposed to the written rules they were viewed as in the past. Since a developer now has no way to determine precisely how the processor will operate, the act of hand-tuning an application at the assembly level is no longer a straightforward task.

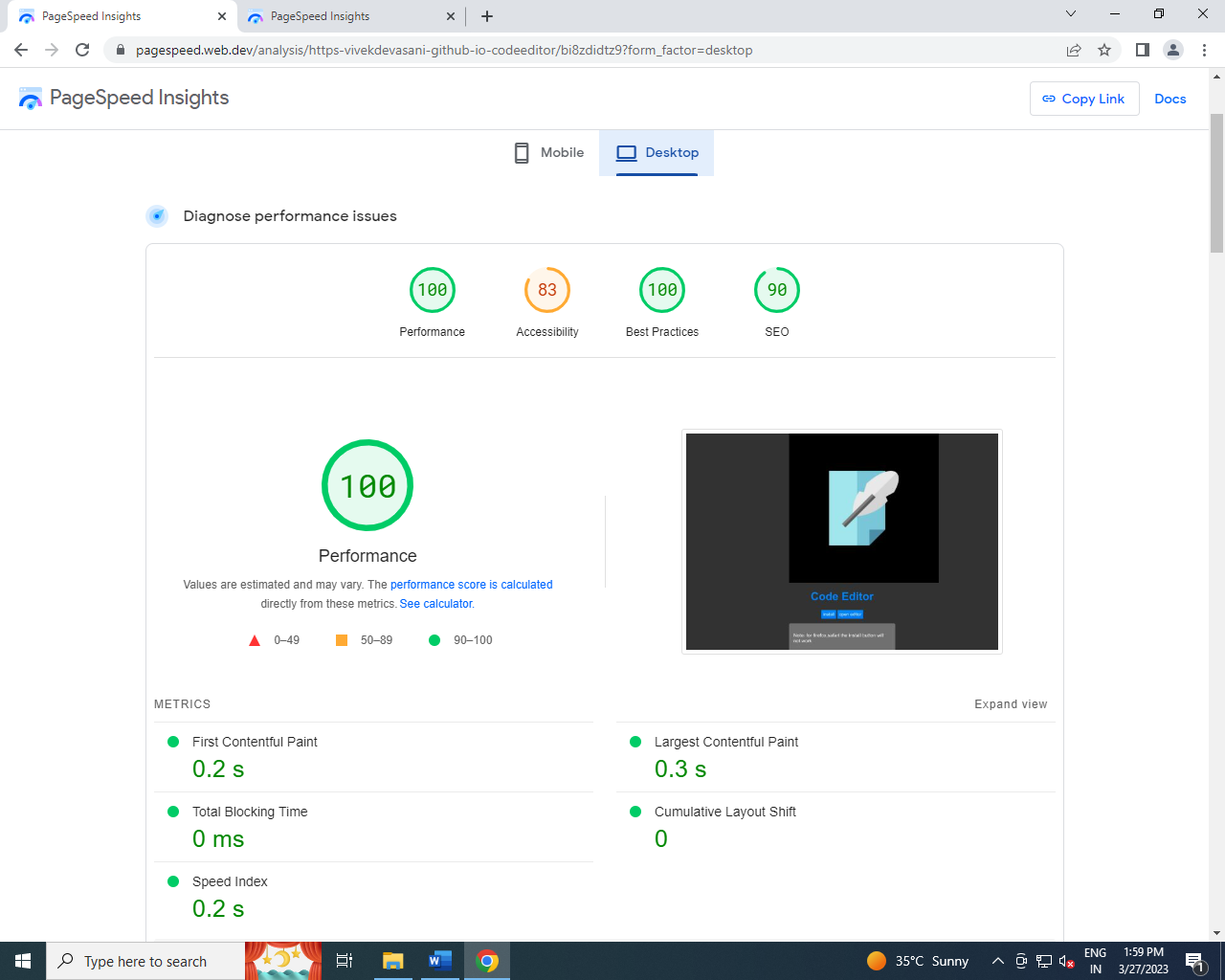
As early as the 1980s, researchers were aware of these trends in computing and began developing tools to automate the task of performance analysis. These initial tools, much like the computers of the time, were simple in nature and capable of gathering only rudimentary performance statistics. Furthermore, due to technical limitations, these early tools focused exclusively on quantifying application-level performance, and were unable to characterize the effects of items such as library code or the operating system itself. As computers became more complex, however, advanced tools were developed to cut through the layers of abstraction caused by the use of advanced operating systems and other development frameworks to gain meaningful performance statistics for the entire software system. More recently, hardware designers have begun to embed counters in the CPU that can record cache hit statistics and other meaningful information, thus allowing developers to obtain a complete performance profile for their applications.

In the remainder of this paper, the various types of performance analysis tools currently available to developers will be examined in-depth with particular attention paid to real-world implementations. Static tools, which make use of counting and/or sampling methods to obtain rudimentary statistics, are treated first. This is followed by an overview of dynamic tools, which utilize binary instrumentation and probing to provide better insight into application performance. Finally, information on hybrid tools that combine both static and dynamic analysis techniques is provided along with a set of closing remarks.

|  |  |  |
| --- | --- | --- |
| **Type** | **Mobile** | **Desktop** |
| **performance** | 81 | 100 |
| **Accessibility** | 83 | 83 |
| **Best practices** | 100 | 100 |
| **SEO** | 92 | 90 |

Perfromance report out of 100

Report from: https://pagespeed.web.dev/analysis/https-vivekdevasani-github-io-codeeditor/1hapyhlj65?form\_factor=desktop

****

**9.FUTURE SCOPE**

Every software designed and developed should be updated with the time to make it adopt changes in the world. As the code development becomes more and more complex we need to add some new features in the code editor.

These new features in the code editor are the adding user custom extensions to the editor. The Code editor extension extends the code editor advanced editing toolbar with an embedded Ace editor widget, providing some handy features for user/site JavaScript pages, CSS pages, JSON pages, and when extension is also installed, for Lua pages, i.e. pages in the Module namespace Which will help the programmer to increase their productivity more. We can also add the emmet extension support to them by default in the HTML mode. This extension increases the productivity and decreases the efforts to write code manually line by line.

The extensions are to be added to the editor with the taking measures. So the extensions will not damage or crash the editor.

We can also add another feature which is adding the custom snippets to the language. A code Snippet is a programming term that refers to a small portion of re-usable source code, machine code, or text. Snippets help programmers reduce the time it takes to type in repetitive information while coding. Code Snippets are a feature on most text editors, code editors, and IDEs.

This is the future scope of the code editor. The above mentioned updates will help the programmers to improve productivity as well as the features of the code editor.

**10. CONCLUSION**

The main objective of the code editor is to make the code editing easy for the programmers in the low-end devices and in mobiles without having the bugs. The first objective is to make editor available for low end devices. The second objective is to have no issues with handling large code where most of the editors face a major problem.

From the above mentioned objectives have prepared the editor which is robust to code handling errors and lightweight with cross platform support. We have given the features like syntax highlighting ,auto code completion to editor which increases the productivity of the programmer. The editor also supports many themes which the programmer can choose as per his comfort. Different editors use different keyboard shortcuts when migrating to a new editor he need to see all of them for new editor. But for our editor we need to the keybindings to have same shortcut keys for functionalities.

The code editor has some limitations some them are when we try to use the editor in the safari browser it limits the programmer to edit only the programs upto 1GB of size and the assistive touch will not work in the desktops as it needs the touch events to move the assistive touch position.

**11. BIBILOGRAPHY**

1) w3schools Website – https://www.w3schools.com

2) pouchdb documentation – https://pouchdb.com

3) MDN Web Docs – https://developer.mozilla.org

4) wed.dev – https://web.dev

5 Carlos Delgado, article on code editors for javascript -https://ourcodeworld.com/articles/read/309/top-5-best-code-editor-plugins-written-in-javascript

6) Sahil Pandita, Aswanth Surendran, Rishiraj Thadeshwar,Ashish Nahak, Prof.Ujwala Gaikwad, “BROWSER BASED CODE EDITOR,” *International Research Journal of Engineering and Technology (*Volume: 08 on 05 May 2021).

7) “Client-Side Data Storage” book by Raymond Camden (e-book)