# **Abstract**



Fig.

Remote desktop is the ability to connect with and use a far away desktop computer from a separate computer. Remote desktop users can access their desktop, open and edit files, and use applications as if they were actually sitting at their desktop computer. Employees often use remote desktop software to access their work computers when they are traveling or working from home.

Remote desktop access is very different from [cloud computing](https://www.cloudflare.com/learning/cloud/what-is-the-cloud/), even though both allow employees to work remotely. In cloud computing, users access files and applications that are stored in the cloud — specifically, in cloud servers. In contrast, when using remote desktop software, users are actually accessing their physical desktop computer, and can only use files and applications saved locally on that desktop. Cloud computing is sometimes easier to use and more efficient to implement for [remote workforces](https://www.cloudflare.com/learning/access-management/remote-workforce-security/), but many companies have not migrated to the cloud, or cannot for security or regulatory reasons.

**What is the Remote Desktop Protocol (RDP)?**

The Remote Desktop Protocol (RDP) is a [protocol](https://www.cloudflare.com/learning/network-layer/what-is-a-protocol/), or technical standard, for using a desktop computer remotely. Remote desktop software can use several different protocols, including RDP, Independent Computing Architecture (ICA), and virtual network computing (VNC), but RDP is the most commonly used protocol. RDP was initially released by Microsoft and is available for most Windows operating systems, but it can be used with Mac operating systems too.

**How does RDP work?**

Think of a remote-controlled drone or toy car. The user presses buttons and steers the drone or car from afar, and their commands are transmitted to the vehicle. Using RDP is somewhat like that: the user's mouse movements and keystrokes are transmitted to their desktop computer remotely, but over the Internet instead of over radio waves. The user's desktop is displayed on the computer they are connecting from, just as if they were sitting in front of it.

The RDP protocol opens a dedicated network channel for sending data back and forth between the connected machines (the remote desktop and the computer currently in use). It always uses network [port](https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/) 3389 for this purpose. Mouse movements, keystrokes, the desktop display, and all other necessary data are sent over this channel via [TCP/IP](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/), which is the transport protocol used for most types of Internet traffic. RDP also [encrypts](https://www.cloudflare.com/learning/ssl/what-is-encryption/) all data so that connections over the public Internet are more secure.

Because keyboard and mouse activity has to be encrypted and transmitted over the Internet, which takes a few milliseconds, and because the desktop display has to be transmitted back to the user, slight delays often occur. For instance, if a user double-clicks on an application to open it, the "double click" may not take place for a few milliseconds as the user's action is transmitted to the desktop before being carried out. Then, when the application opens, there may be another short delay while the display is transmitted back to the user.

# *Preface*

# 1.1 INTRODUCTION

One of the first barriers to overcome before contributing to a project is configuring your local development environment. The time-consuming task of managing related dependencies and troubleshooting incompatible versions can be discouraging, especially for someone working who contributes infrequently or works on multiple projects. In more advanced environments, a development team may even have a standard set of tools, extensions, and configuration files, adding to the delicate nature of the development environment.

Developers want to spend less time managing their development environment and more time contributing high-quality code.

Eliminating the responsibility of configuring and maintaining a local development environment frees up valuable development time and streamlines on boarding of new developers joining the team.

Remote Development, as the name suggests, directly targets software developers and those who manage or support teams of developers. However, a mature remote development offering lowers the technical barrier to collaboration and enables non-developer personas to effectively and efficiently contribute.

Remotely Visible IDE makes it easy control and file transfer functionality. With a Remotely Visible IDE, users are able to Connect from computers sharing features, so that it becomes easy for client- server interaction.

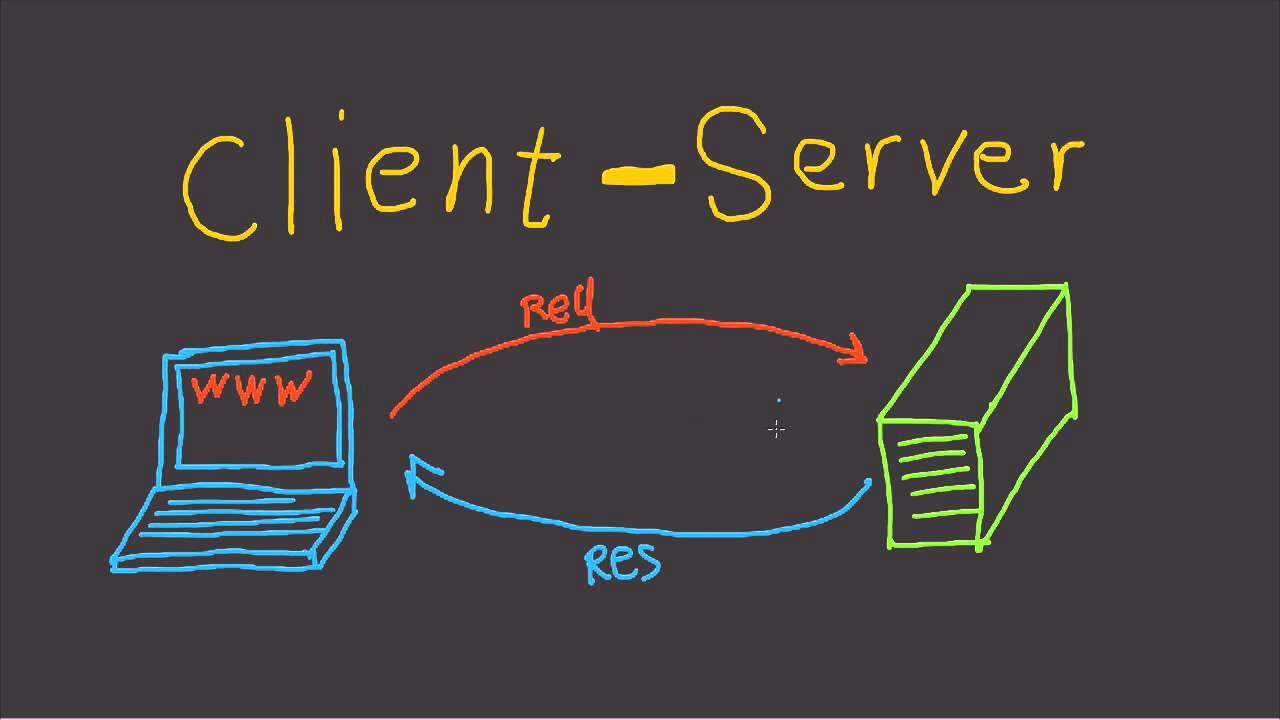


Fig.

Through ****Remote Desktop Connection**** tool will help you in this situation where you can access the required PC from afar. The computer just needs to be turned on and set up for remote connection and you are good to go.

You can do all sorts of things, open a file, run an application or just remotely work on it.

# 1.2 Motivation of the project

The motivation behind creating a remote IDE (Integrated Development Environment) is to provide developers with a centralized platform for coding, testing, and deploying their applications. With a remote IDE, developers can access all the necessary tools and resources required for software development from anywhere at any time. This allows for greater flexibility and collaboration between team members who may be geographically dispersed.

Additionally, a remote IDE can provide cost savings by reducing the need for physical hardware and infrastructure, as all resources are provided through the cloud. It also enables developers to work on different operating systems and devices without needing to install specific software or tools locally.

Overall, a remote IDE can improve developer productivity, streamline workflows, and enhance collaboration among team members, making it an attractive option for modern software development teams.

**1.3 Basic Description of the project**

The remote IDE project is a software development platform that allows developers to code, test, and deploy their applications from anywhere at any time. It is a centralized platform that provides all the tools and resources necessary for software development, including an integrated code editor, debugging tools, version control systems, and project management tools.

The remote IDE is hosted in the cloud, making it easily scalable and accessible from anywhere with an internet connection. Developers can access their codebase and work on it collaboratively with other team members, regardless of their location. The cloud-based infrastructure also enhances security, reducing the risk of data breaches and cyber threats.

One of the key benefits of the remote IDE is its built-in version control system, which tracks changes to code and enables team members to collaborate more efficiently. This helps to improve code quality and reduce errors and bugs. Additionally, the remote IDE can integrate with other tools and services, such as issue trackers and project management platforms, to provide a complete software development lifecycle solution.

Overall, the remote IDE project is designed to help organizations improve their software development workflows, enhance collaboration between team members, and ultimately build better products faster.

1. **Features**

A remote IDE (Integrated Development Environment) is a web-based application that allows developers to write, edit, and debug their code remotely. Some features of a remote IDE include:

**1. Code editing:** A remote IDE provides code editing tools such as syntax highlighting, code completion, and automatic indentation.

**2. Collaboration:** Multiple developers can work on the same project simultaneously using a remote IDE. This collaboration feature includes real-time code sharing and commenting.

**3. Debugging:** Remote IDE offer debugging features such as breakpoints, step-by-step execution, and live variable inspection.

**4. Version control integration:** Remote IDE integrate with version control systems like Git, allowing developers to manage their code changes easily.

**5. Build and deployment tools:** Remote IDE provide build and deployment tools to help developers automate the deployment process for their applications.

**6. Customization options:** Developers can customize their remote IDE by adding plugins, themes, and other extensions to enhance their coding experience.

**7. Cloud-based storage:** Remote IDE offer cloud-based storage solutions that allow developers to save their work in the cloud and access it from anywhere with an internet connection.

**8.Integrated terminal:** Remote IDE often have an integrated terminal that allows developers to run commands directly within the IDE.

**9.Multiple language support**: Remote IDE support multiple programming languages, allowing developers to work on different projects without switching between multiple IDE.

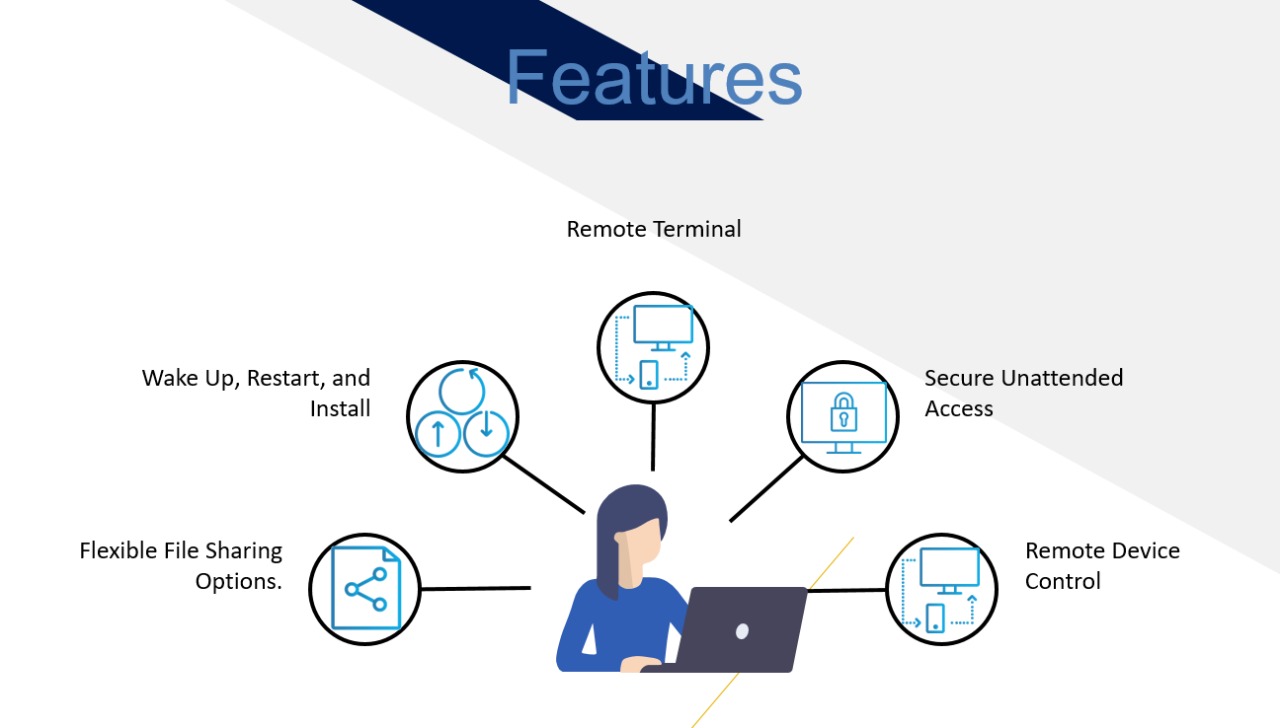


Fig.

**3. Project Details**

Remotely visible IDE is a remote desktop application . The proprietary software program provides platform independent remote access to personal computers and other devices running the host application. It offers remote control and file transfer functionality. It is one of many tools that have been used in technical support . Ide is a fast and secure all-in-one solution for gaining access to computers and within the networks remotely.

-> Server PC can view desktop of clients PCs

-> Server PC can control clients PCs with servers mouse and keyboard.

# ***Overview***

Remote development lets you use the IDE interface on a thin client while having a powerful remote host to check out and load your project, index, analyze, build, run, debug, and test your code.

The remote host is a physical (Linux) or virtual machine hosting the source code and running a headless instance. The thin client, for example, a laptop, connects to the host and transparently provides full access to all IDE features.

* 1. ***Workflows***

There are two types of main workflows available in remote development:

**-> Server-to-client flow:** when an organization first sets up a remote headless server, obtains a link (customized or not) which then can be used by you on Client (for example, your laptop) to connect to the available server.

**-> Client-to-server flow:** when you set up and install the remote server from the Client UI using the SSH or other connection.

Connect to a remote project at manually launched remote IDE (Server-to-client flow).

If you already have IDE installed on your remote server, you can launch it manually and connect to the remote project started in that IDE. It works the same way as from [JetBrains Gateway](https://www.jetbrains.com/help/idea/remote-development-a.html" \l "gateway).

Use this approach if your company has a custom orchestration or in case your remote IDE starts automatically on its side.

The process can be described as follows:

1.Start a backend project in the remote IDE.

2.Select one of the connection links generated by the backend.

3.Open the link on your local machine.

* 1. ***Tools and Technology***
* Java Programming Language
* Java Swing
* Java Robot
* Java Networking
* Socket Programming
* Visual Studio Code

**Java Programming Language :** Java is a high-level, object-oriented programming language that was first introduced in 1995 by Sun Microsystems. It has since become one of the most popular programming languages in the world, thanks to its versatility and ease of use.

One of the biggest advantages of Java is its ability to run on multiple platforms, including Windows, macOS, Linux, and even mobile devices such as Android phones. This makes it an ideal choice for developing cross-platform applications and software.

Java is also known for its robustness and security features. Its automatic memory management system ensures that memory leaks and other common programming errors are minimized, while its strict type checking and exception handling help to prevent runtime errors. Additionally, Java's sandbox security model prevents untrusted code from accessing sensitive resources on the user's computer or network.

Another key feature of Java is its support for multithreading, which allows developers to write code that can execute multiple tasks simultaneously. This is particularly useful for building applications with complex user interfaces or that require background processing, such as web servers or database management systems.

Java's syntax is similar to that of C++, making it relatively easy for programmers who are familiar with that language to learn Java. However, Java also includes many features that simplify common programming tasks, such as garbage collection, which eliminates the need for manual memory management.

Java is also widely used in enterprise environments, where it forms the backbone of many large-scale applications and systems. This is due in part to its support for distributed computing, which allows Java applications to communicate with each other over a network, even across different hardware and operating systems.

One of the most popular uses of Java is for building web applications, using frameworks such as Spring, Hibernate, and Struts. These frameworks provide a set of tools and libraries that simplify common web development tasks, such as handling HTTP requests and responses, managing sessions and cookies, and connecting to databases.

Java is also commonly used for developing mobile applications, particularly for Android devices. The Android operating system is built on top of a Java-based framework called the Android Runtime, which allows developers to write native Android apps using Java.

Overall, Java's popularity can be attributed to its wide range of features and versatility. It offers a powerful set of tools for building everything from desktop applications to large-scale enterprise systems, and its ability to run on multiple platforms makes it an excellent choice for cross-platform development. Its robustness and security features also make it a popular choice for building mission-critical applications and systems.

**Java Swing :** Java Swing is a powerful graphical user interface (GUI) toolkit that is used to create desktop applications in Java. It provides developers with a variety of components and tools that can be used to create complex and dynamic GUIs for their applications.

Swing was introduced as a replacement for the older Abstract Window Toolkit (AWT) in Java 1.2, and it quickly became the preferred choice for creating GUIs due to its rich feature set and cross-platform compatibility.

One of the key benefits of using Swing is that it allows developers to create GUIs that have a consistent look and feel across different platforms, such as Windows, MacOS, and Linux. This is achieved through the use of a pluggable look and feel architecture that allows developers to customize the appearance of their applications without having to worry about platform-specific details.

Swing also provides a wide variety of components that can be used to create complex and interactive GUIs. These include buttons, labels, text fields, combo boxes, check boxes, radio buttons, sliders, progress bars, and many more. In addition, Swing provides support for advanced components such as tables, trees, and tabbed panes, which are commonly used in enterprise applications.

Another important feature of Swing is its support for event-driven programming. This means that developers can write code that responds to user actions, such as button clicks or menu selections. This allows for highly interactive and responsive GUIs that can handle a wide range of user inputs.

Swing also provides a range of layout managers that can be used to position and organize GUI components on the screen. These managers allow developers to create complex layouts that adjust dynamically based on the size of the window or the contents of the components.

One of the strengths of Swing is its ability to provide developers with a high degree of flexibility and control over the appearance and behavior of their applications. This is achieved through the use of various customization options, including the ability to set fonts, colors, borders, and other visual properties of GUI components.

Swing also provides a powerful graphics API that can be used to create custom graphics and animations within GUIs. This allows developers to create dynamic and engaging interfaces that are visually appealing and highly interactive.

In addition to its core set of features, Swing also provides a wide range of tools and libraries that can be used to extend its capabilities even further. For example, the Java 2D API can be used to create advanced graphics and animations, while the JavaFX library provides support for creating modern, web-style GUIs.

Overall, Java Swing is a powerful toolkit that provides developers with a wide range of tools and components for creating rich and dynamic desktop applications in Java. Its cross-platform compatibility, event-driven programming model, and customization options make it an ideal choice for developing enterprise applications that require complex and interactive user interfaces.

**Java Robot:** Java Robot is a powerful class in the Java programming language that allows developers to simulate user input and control the keyboard and mouse programmatically. The Robot class is part of the java.awt package, which provides support for creating graphical user interfaces (GUIs) in Java.

The Robot class can be used to automate repetitive tasks, simulate human input, and test GUI applications. It can also be used to create custom tools and utilities that interact with other applications or services.

One of the key features of the Robot class is its ability to simulate user input events such as mouse clicks, key presses, and mouse movements. With this feature, developers can create scripts or programs that can perform tasks such as filling out forms automatically, clicking buttons, navigating menus, and more.

In addition to simulating user input, the Robot class can also be used to capture screenshots of the screen and manipulate pixels on the screen. This feature is useful for creating automated tests that verify the correctness of GUI components and their behavior under different conditions.

To use the Robot class, developers must first create an instance of the class using the default constructor or one of its overloaded constructors. Once an instance has been created, developers can call methods such as mouseMove(), mousePress(), mouseRelease(), keyPress(), keyRelease(), and more to simulate user input events.

For example, the following code snippet demonstrates how to use the Robot class to move the mouse cursor to a specific position on the screen:

Another useful feature of the Robot class is its ability to generate keyboard events. Developers can use the keyPress() and keyRelease() methods to simulate key presses and releases, respectively. For example, the following code snippet demonstrates how to simulate the pressing of the "Enter" key:

In addition to simulating user input events, the Robot class can also be used to capture screenshots of the screen. The createScreenCapture() method can be used to capture a rectangular region of the screen and return it as an image. For example, the following code snippet demonstrates how to capture a screenshot of the entire screen:

Overall, the Java Robot class is a powerful tool that allows developers to simulate user input, control the keyboard and mouse programmatically, and automate repetitive tasks. Its ability to capture screenshots and manipulate pixels on the screen makes it useful for creating automated tests and custom tools and utilities.

**Java Networking :** Java networking is a powerful aspect of the Java programming language that allows developers to create networked applications and services. The Java platform provides a rich set of classes and interfaces for creating and managing network connections, sending and receiving data over the internet, and implementing network protocols and services.

One of the key benefits of using Java for network programming is its platform independence. Java programs can run on any platform that supports the JVM (Java Virtual Machine), which makes it easy to write network applications that work across different operating systems and devices.

The Java networking API is built on top of the TCP/IP stack, which is the underlying protocol used for communication over the internet. The API provides a high-level abstraction layer that makes it easy for developers to create networked applications without having to worry about the low-level details of network communication.

The core package for Java networking is java.net, which contains classes and interfaces for creating and managing network connections, sending and receiving data over the network, and working with network protocols such as HTTP, FTP, and more.

One of the key classes in the java.net package is the Socket class, which represents a network socket connection. A socket is an endpoint for sending and receiving data over a network. With the Socket class, developers can create client and server applications that communicate over a network.

This code will establish a connection to the web server at www.example.com on port 80, send a HTTP request to retrieve the webpage at /index.html, and print out the response to the console.

Another important class in the java.net package is the ServerSocket class, which represents a server-side socket that listens for incoming connections. With the ServerSocket class, developers can create server applications that accept client connections and communicate with them over a network.

This code will create a server socket on port 8080 and wait for incoming connections. When a client connects, it will read the HTTP request, generate a response that contains a simple HTML page, and send it back to the client.

Java networking also provides support for working with higher-level protocols such as HTTP, FTP, SMTP, and more. For example, the java.net.HttpURLConnection class can be used to send HTTP requests and receive responses from web servers, while the java.net.FTPClient class can be used to connect to an FTP server and transfer files.

Overall, Java networking is a powerful aspect of the Java programming language that allows developers to create networked applications and services. Its rich set of classes and interfaces provide support for creating and managing network connections, sending and receiving data over the internet, and implementing network protocols and services.

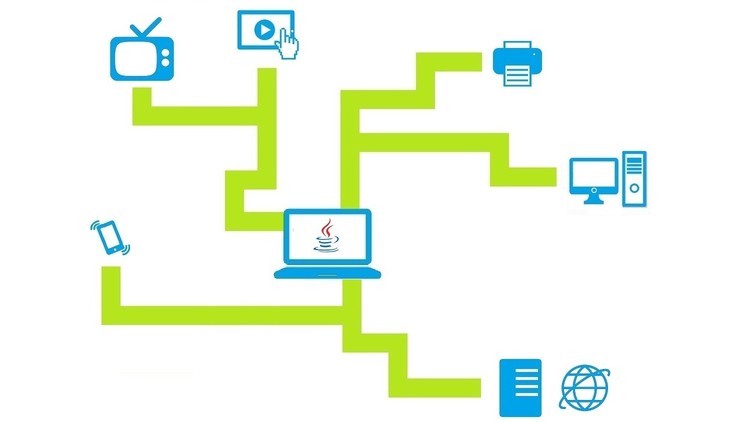


Fig.

**Socket Programming:** Socket programming is a way of communicating between different processes on different systems. It allows processes to send and receive data over a network using sockets, which are endpoints that enable communication between two applications.

Sockets can be used to create both client-side and server-side applications. The client-side application creates a socket and connects it to a server-side socket. Once connected, the client can send and receive data to and from the server. The server-side application listens for incoming connections, accepts the connection requests, and then communicates with the clients by sending and receiving data.

The basic steps involved in socket programming are as follows:

1. Create a socket: A socket is created using the socket() system call. This call returns a file descriptor that represents the socket and will be used in subsequent system calls.

2. Bind the socket: The bind() system call binds the socket to a specific address and port number on the local machine. This step is done only in the case of servers; clients don't need to bind their sockets.

3. Listen for incoming connections: Servers use the listen() system call to listen for incoming connection requests.

4. Accept connections: When a client sends a connection request, the accept() system call is used to accept the connection and create a new socket for communication with that client.

5. Send and receive data: Data can be sent and received using the send() and recv() system calls.

6. Close the socket: Once the communication is complete, the close() system call is used to close the socket.One of the most important concepts in socket programming is the use of protocols. A protocol is a set of rules that govern how data is transmitted and received over a network. There are many different protocols available, including TCP and UDP.

TCP (Transmission Control Protocol) is a reliable and connection-oriented protocol. It guarantees that all data sent will arrive at its destination and in the correct order. TCP is commonly used for applications that require reliable and error-free data transfer, such as web browsing and file transfers.

UDP (User Datagram Protocol) is a connectionless protocol that does not guarantee delivery of data or ensure the correct order of data transmission. UDP is useful for applications that need to send small amounts of data quickly, such as online games and chat applications.

Socket programming can be used in a wide range of applications, from simple chat programs to complex distributed systems. It allows developers to create applications that communicate with other applications running on different machines, regardless of the operating system or hardware platform being used.

In conclusion, socket programming is a powerful tool for building networked applications. Understanding how sockets work, and how to use them effectively, is essential for any developer working on networked applications. By following the basic steps outlined above and choosing the appropriate protocol for your application, you can build robust and reliable applications that can communicate over networks with ease.

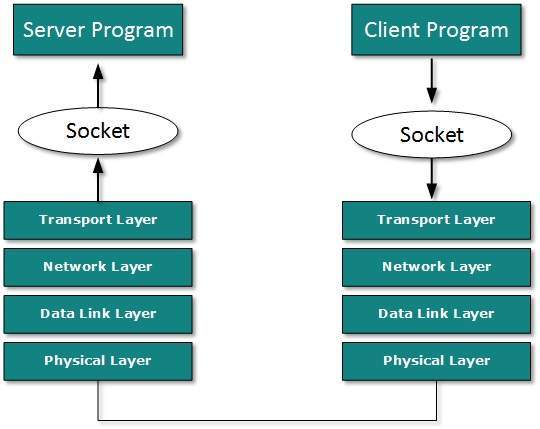


Fig.

The primary socket API functions and methods in this module are:

* socket()
* .bind()
* .listen()
* .accept()
* .connect()
* .connect\_ex()
* .send()
* .recv()
* .close()

Python provides a convenient and consistent API that maps directly to system calls, their C counterparts. In the next section, you’ll learn how these are used together.

As part of its standard library, Python also has classes that make using these low-level socket functions easier. Although it’s not covered in this tutorial, you can check out the [socketserver module](https://docs.python.org/3/library/socketserver.html), a framework for network servers. There are also many modules available that implement higher-level Internet protocols like HTTP and SMTP. For an overview, see [Internet Protocols and Support](https://docs.python.org/3/library/internet.html).

## TCP Sockets

## We are going to create a socket object using socket.socket(), specifying the socket type as socket.SOCK\_STREAM. When you do that, the default protocol that’s used is the [Transmission Control Protocol (TCP)](https://en.wikipedia.org/wiki/Transmission_Control_Protocol). This is a good default and probably what you want.

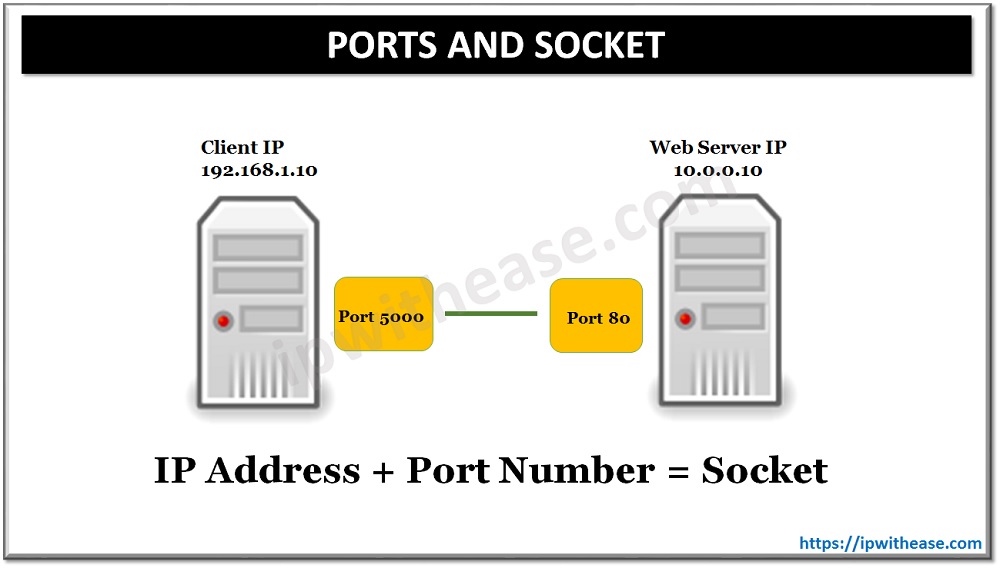
****

Fig.

**TCP Socket flow diagram**

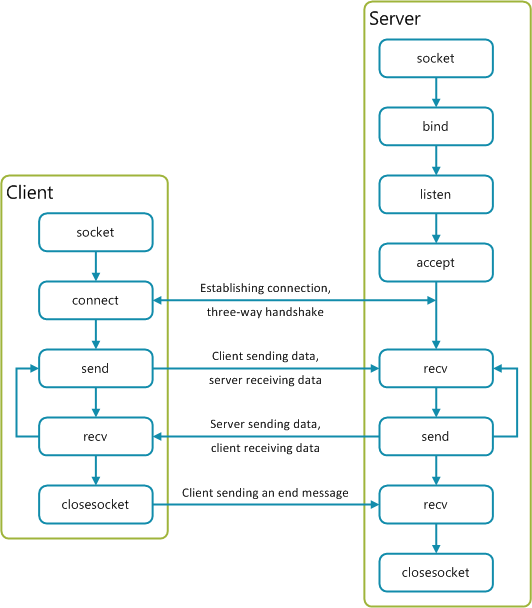


Fig.

**Visual Studio Code:** Visual Studio Code (VS Code) is a code editor developed by Microsoft that has gained immense popularity in the developer community. It is a lightweight, cross-platform IDE (integrated development environment) that provides a customizable and extensible environment for developers to write, debug and test their code. In this essay, we will explore why VS Code is so popular among developers, its features and benefits.

One of the primary reasons for the popularity of VS Code is its versatility. It supports a wide range of programming languages, including but not limited to JavaScript, Python, Java, Ruby, C++, and PHP. Moreover, it provides a vast array of extensions that enable developers to customize and enhance the functionality of the editor to suit their specific needs. These extensions can be installed from the VS Code Marketplace, which offers a large collection of tools and libraries that cater to different aspects of software development.

Another significant advantage of VS Code is its ease of use. The user interface is intuitive and straightforward, making it easy for even novice programmers to get started quickly. Furthermore, VS Code provides a range of features such as syntax highlighting, auto-completion, and debugging that help streamline the coding process. The editor also comes with built-in Git support that enables developers to manage their code repositories directly from within the editor itself.

One of the most useful features of VS Code is its ability to integrate with various tools and services. For example, it integrates seamlessly with Azure cloud services, providing developers with a comprehensive set of tools to build, deploy and manage their applications on the cloud. Additionally, it integrates with other popular tools such as Docker and Kubernetes, allowing developers to build and deploy containerized applications easily.

VS Code also provides a highly customizable environment that enables developers to tailor the editor to their workflow. The editor allows developers to add custom shortcuts, key bindings, and snippets, which can significantly improve productivity. Moreover, the editor supports theming, providing developers with the ability to personalize the look and feel of the editor to their liking.

Another significant advantage of VS Code is its active community. The editor has a vast and active community of developers who continuously contribute to improving the editor's features and functionality. This community develops and maintains a wide range of extensions, themes, and plugins that enhance the capabilities of the editor.

In conclusion, Visual Studio Code has become one of the most popular code editors among developers due to its versatility, ease of use, integrations, customizability, and active community. It offers a comprehensive suite of tools for developers to write, test, and debug their code across different programming languages and platforms. With its vast collection of extensions and plugins, VS Code provides a flexible and adaptable environment that can cater to the unique needs of any developer.

4.**CLIENT- SERVER ARCHITECTURE**

Client-server architecture is a widely adopted model in modern computing systems. It refers to the division of responsibilities between a client machine and a server machine, where the client requests services from the server, and the server provides those services in response.

The client is typically a user-facing machine that runs applications designed to access data or services provided by the server. The server machine, on the other hand, is responsible for managing and processing data requests from clients and delivering results back to the clients. The communication between the client and server is facilitated by a network, which can either be a local area network (LAN) or wide area network (WAN), depending on the location of the client and server machines.

One of the main benefits of the client-server architecture is its simplicity and modularity. The separation of responsibilities between the client and server makes it easier to develop and maintain both components independently. The client can be designed to handle specific user interface functionalities, while the server can focus on processing data and delivering results back to the client. This modular approach allows for scalability and flexibility, as each component can be upgraded or replaced as needed without affecting the other.

Another significant benefit of the client-server architecture is its security. The server is responsible for storing and managing important data, which reduces the risk of data loss or corruption on the client machine. The server can also implement security measures such as authentication and encryption to protect the data from unauthorized access. This way, even if a client machine is compromised, the sensitive data is still secure.

In addition to providing enhanced security, the client-server architecture also allows for centralized management and control. The server can monitor and manage the activities of all connected clients, making it easier to detect and resolve issues as they arise. This centralization also enables administrators to enforce standards across the organization, ensuring consistency and compliance with regulations.

However, there are also some drawbacks to the client-server architecture. One of the most significant disadvantages is the potential for network latency and downtime. Because communication between the client and server is dependent on network connectivity, any issues with the network can result in delays or service disruptions. Additionally, servers are typically more expensive and require more maintenance than client machines, which can increase the overall cost of the system.

Despite these drawbacks, the client-server architecture remains one of the most popular models in modern computing systems. Its simplicity, modularity, security, and centralized management make it an attractive option for organizations of all sizes. As technology continues to evolve, the client-server architecture will likely continue to play a critical role in the development of new computing systems and applications.

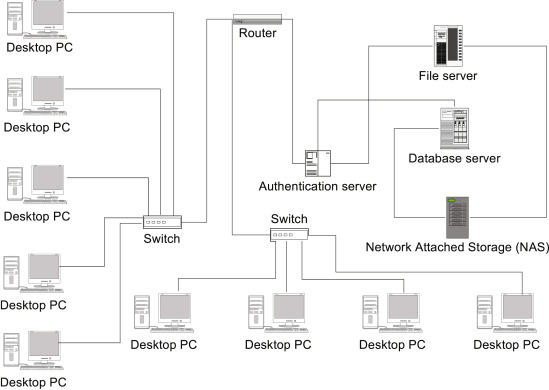
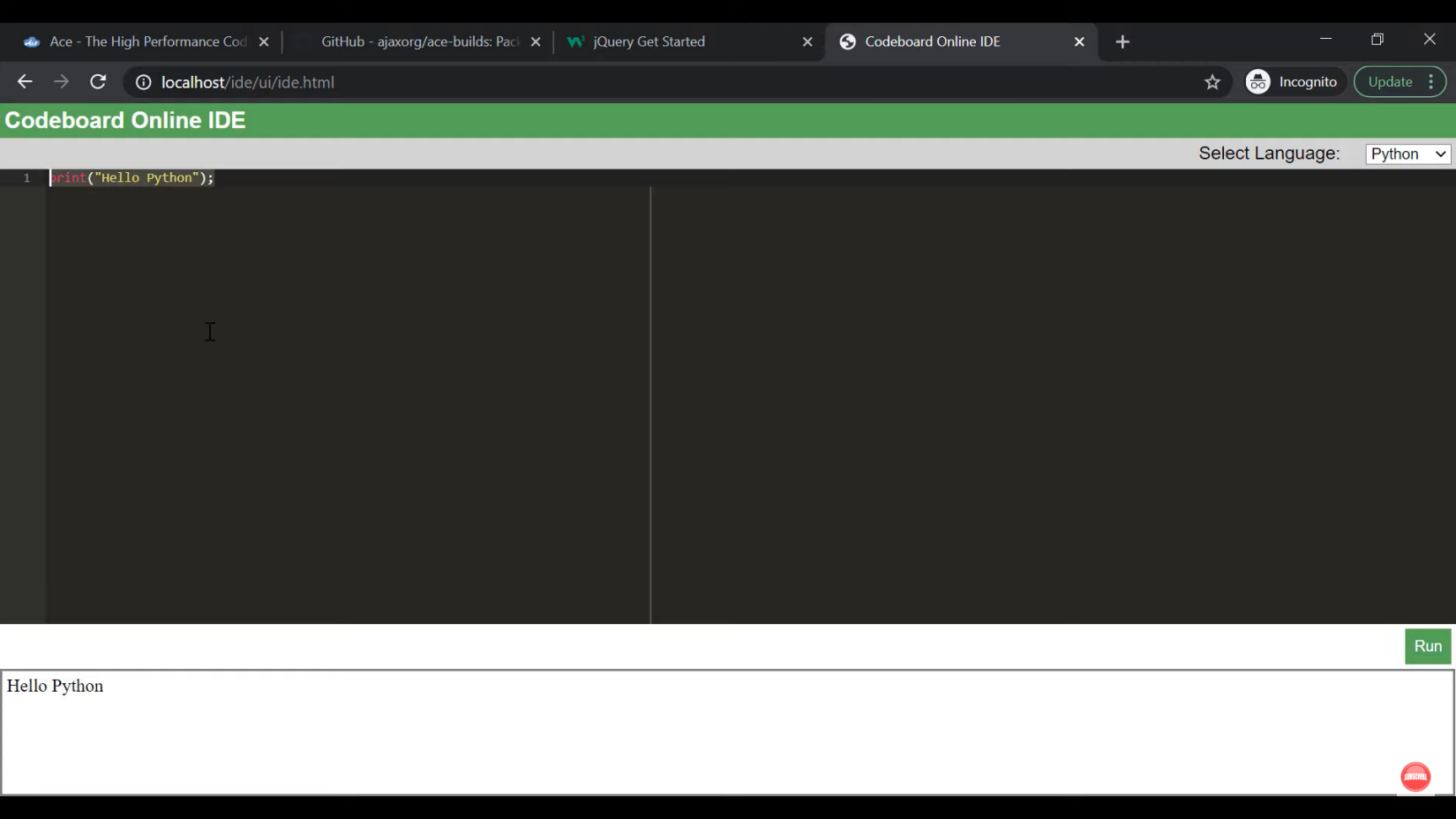
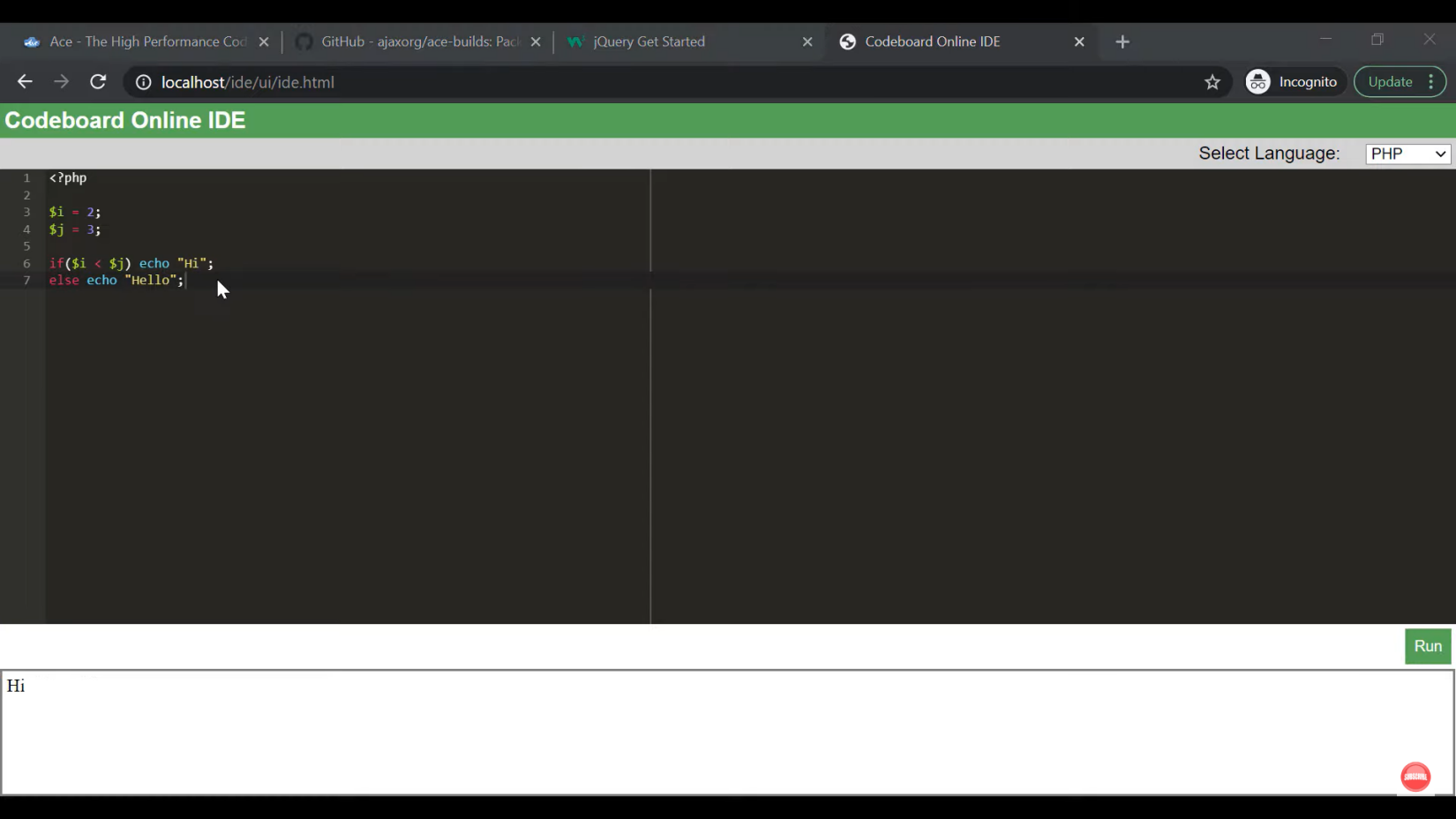
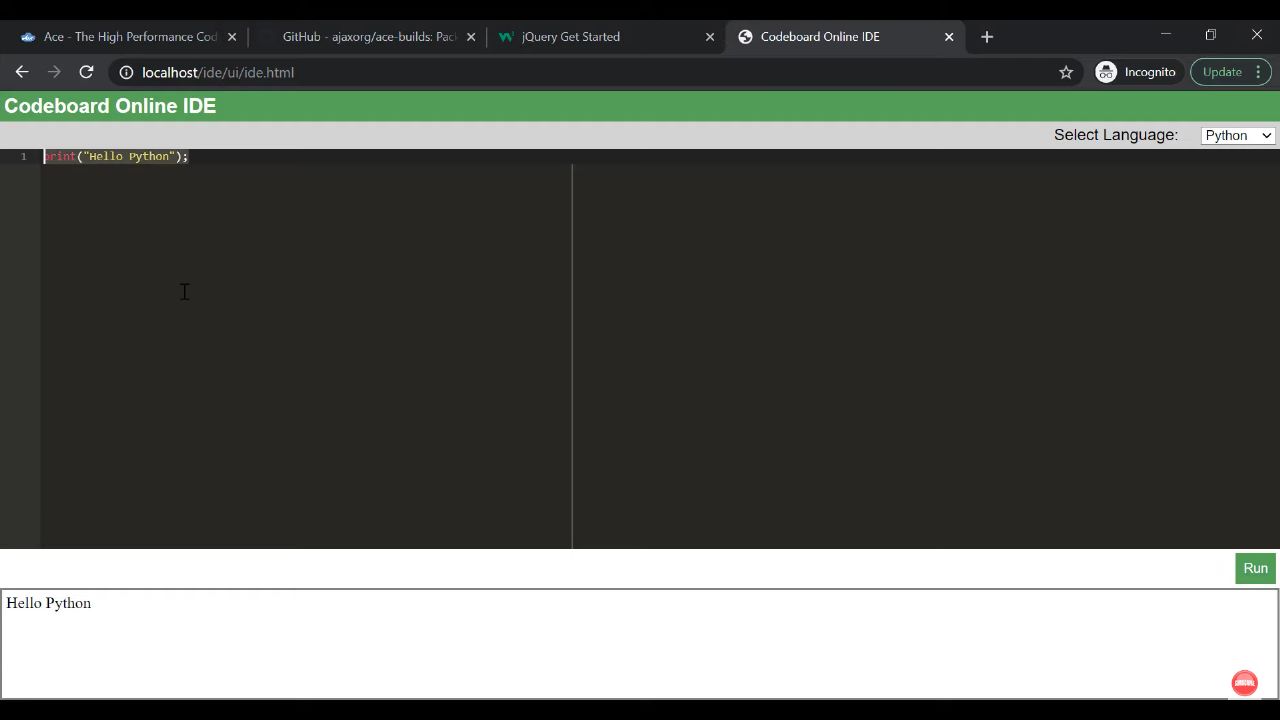
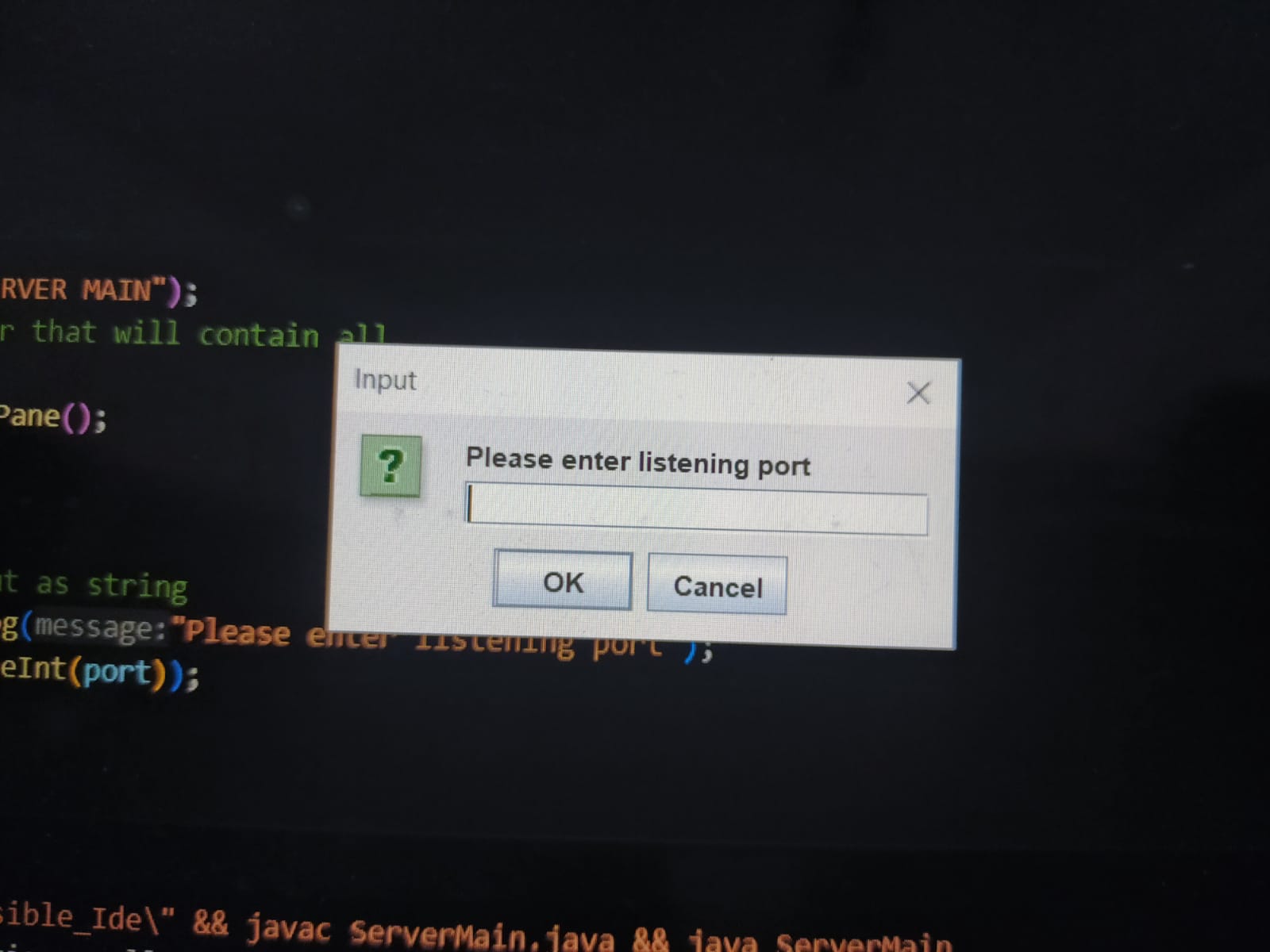
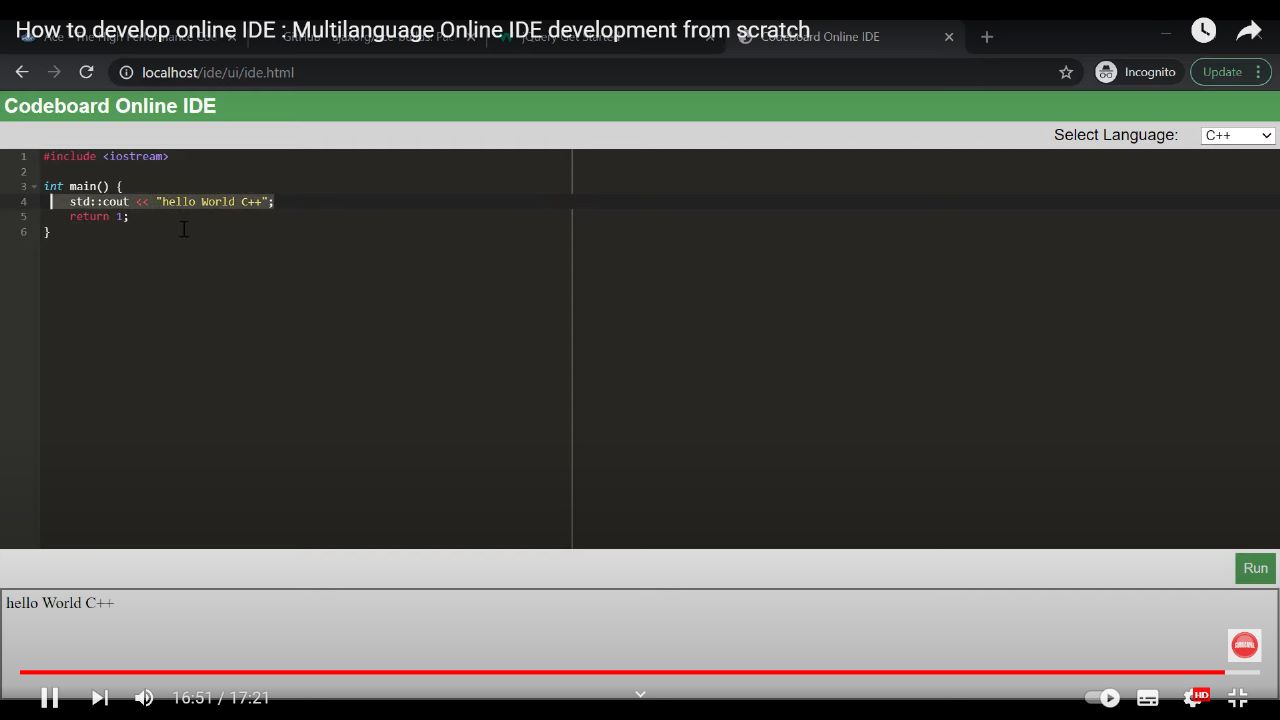
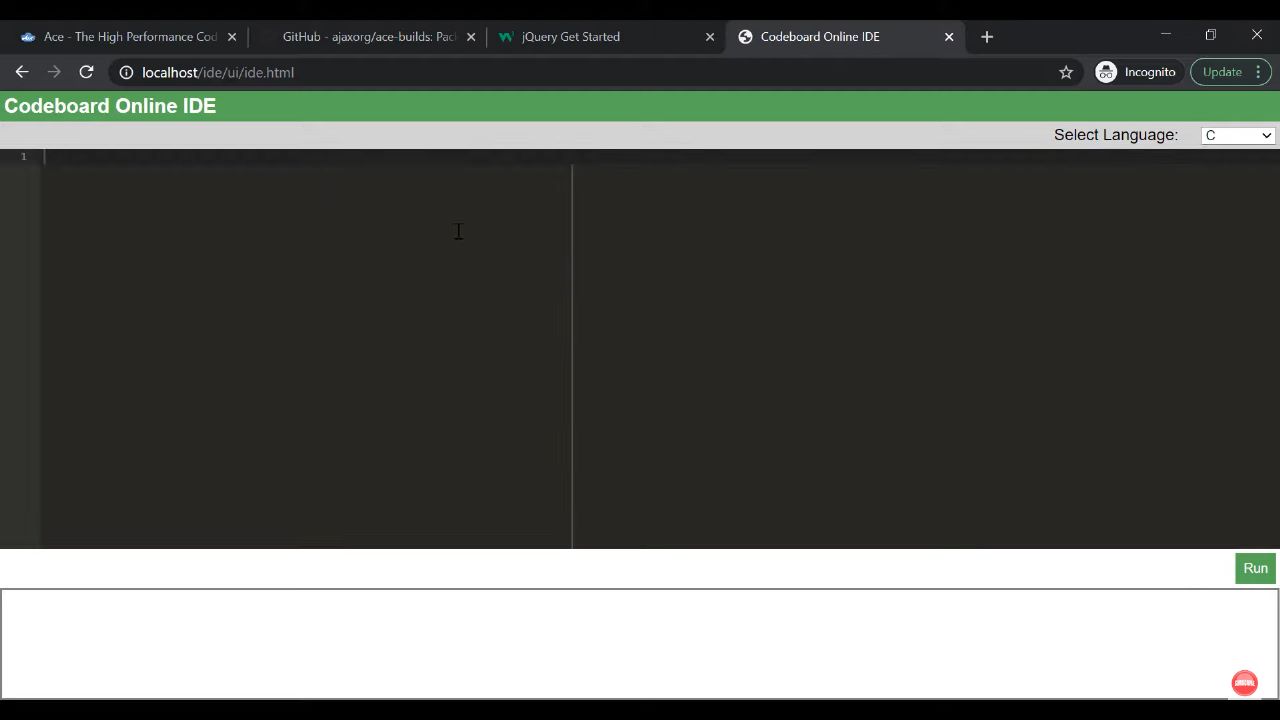
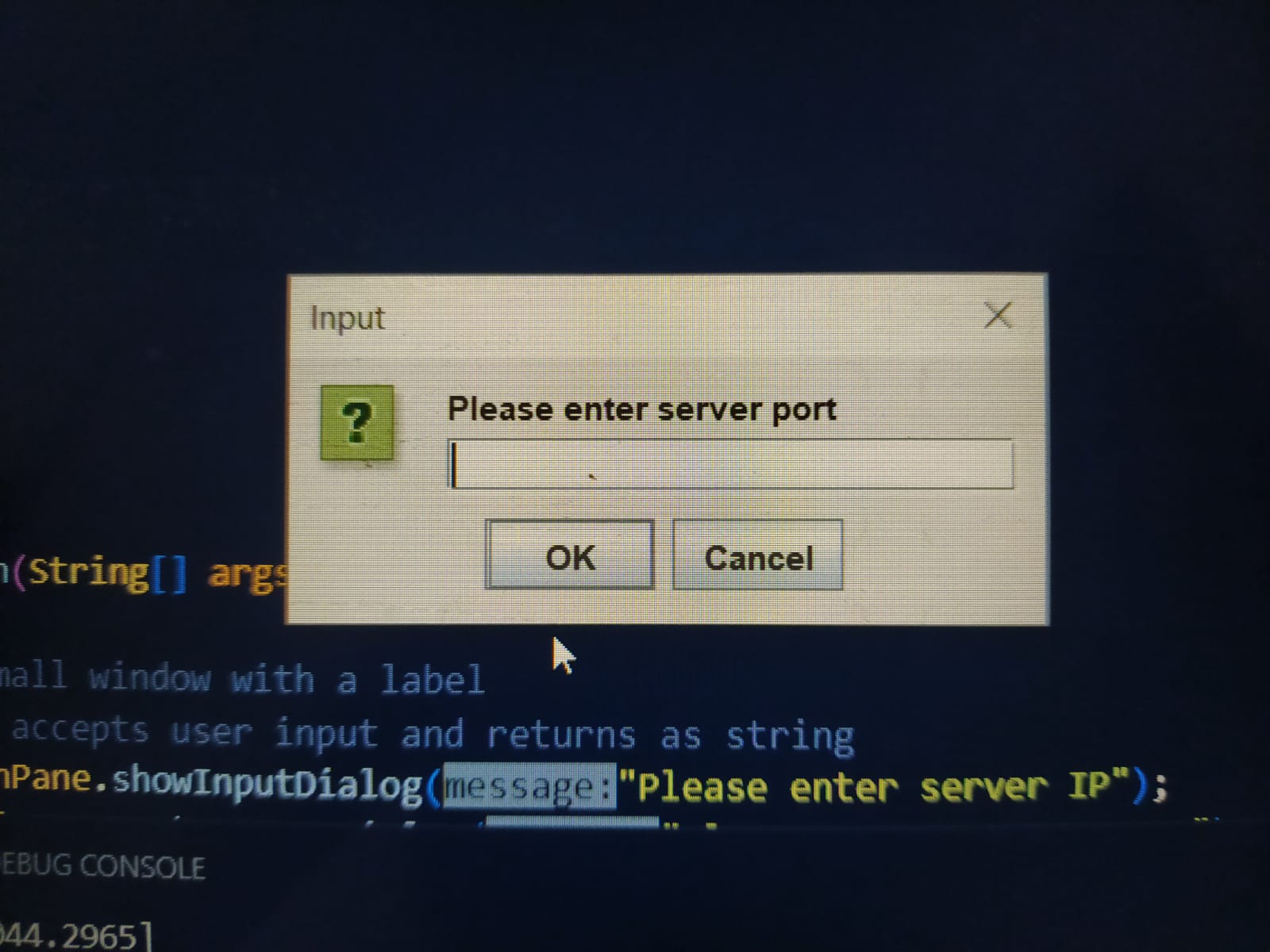
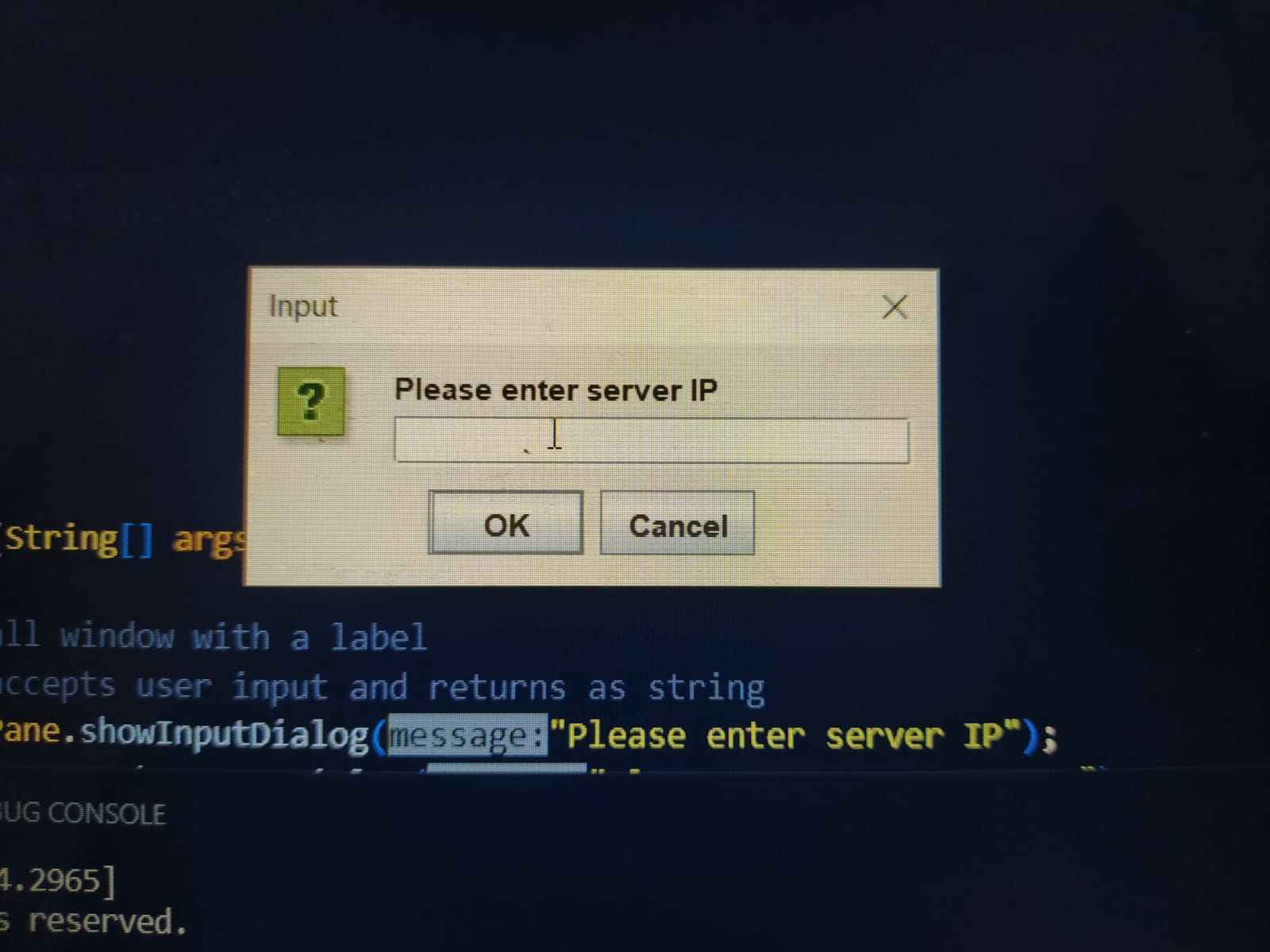
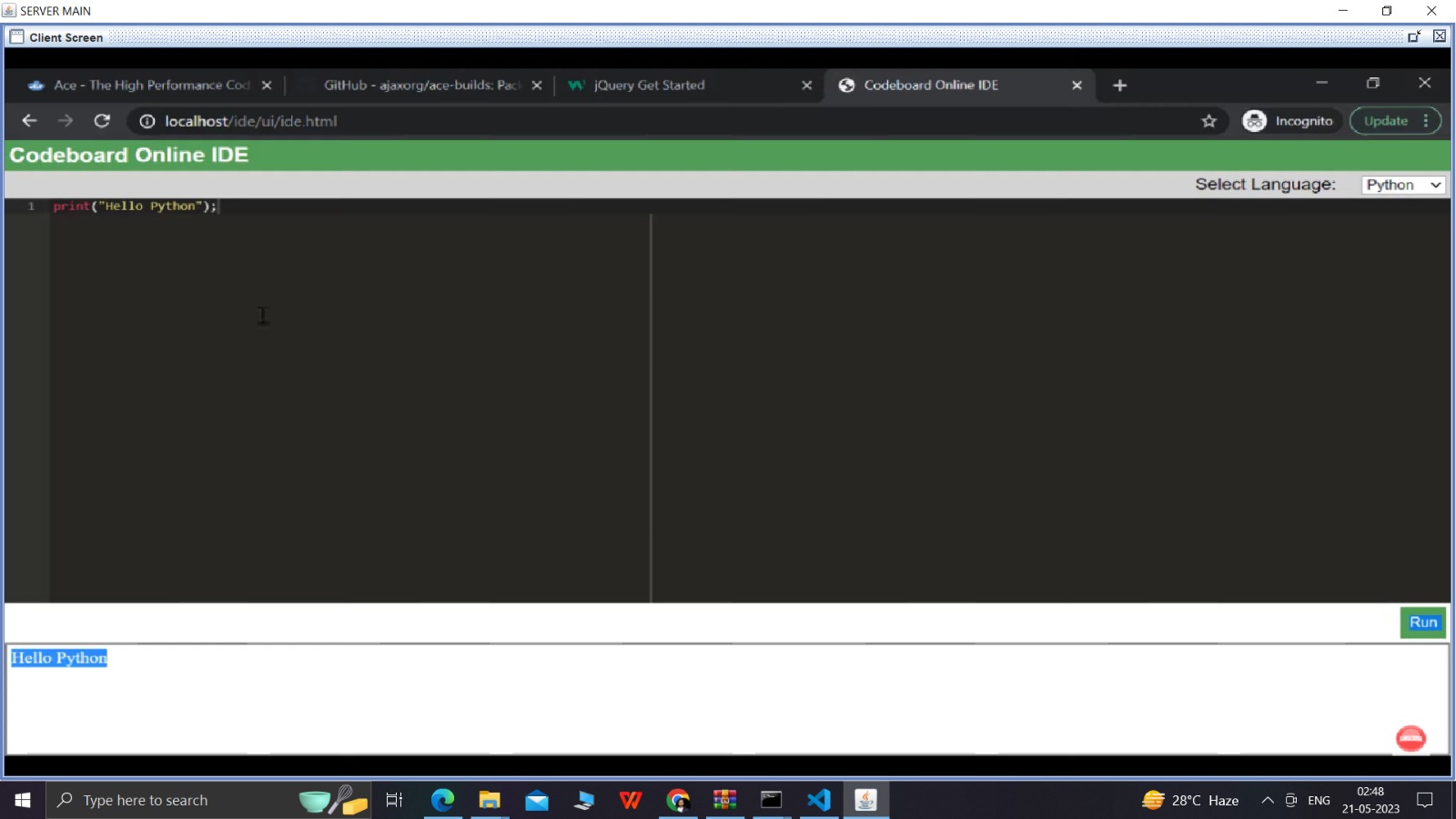
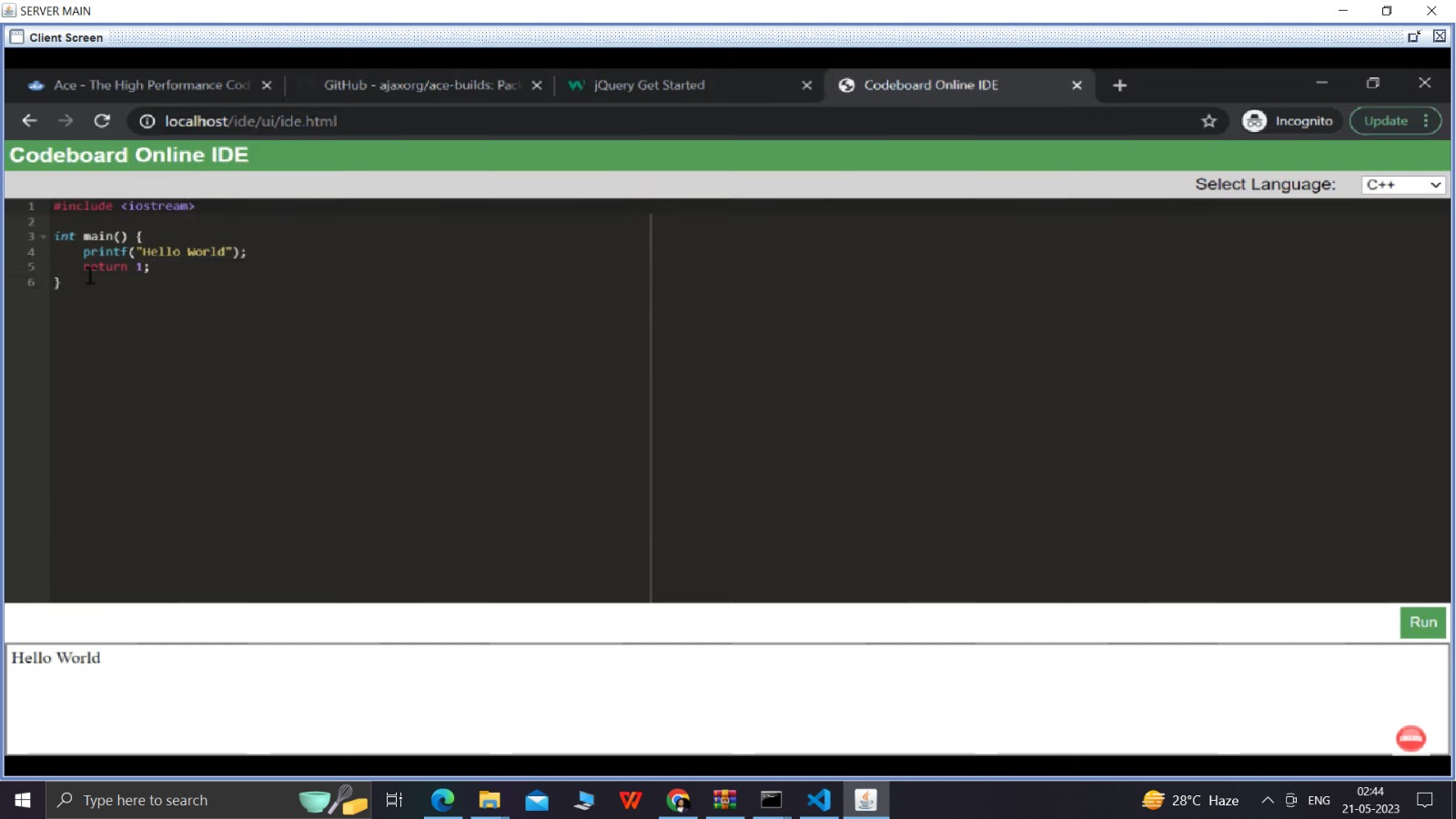
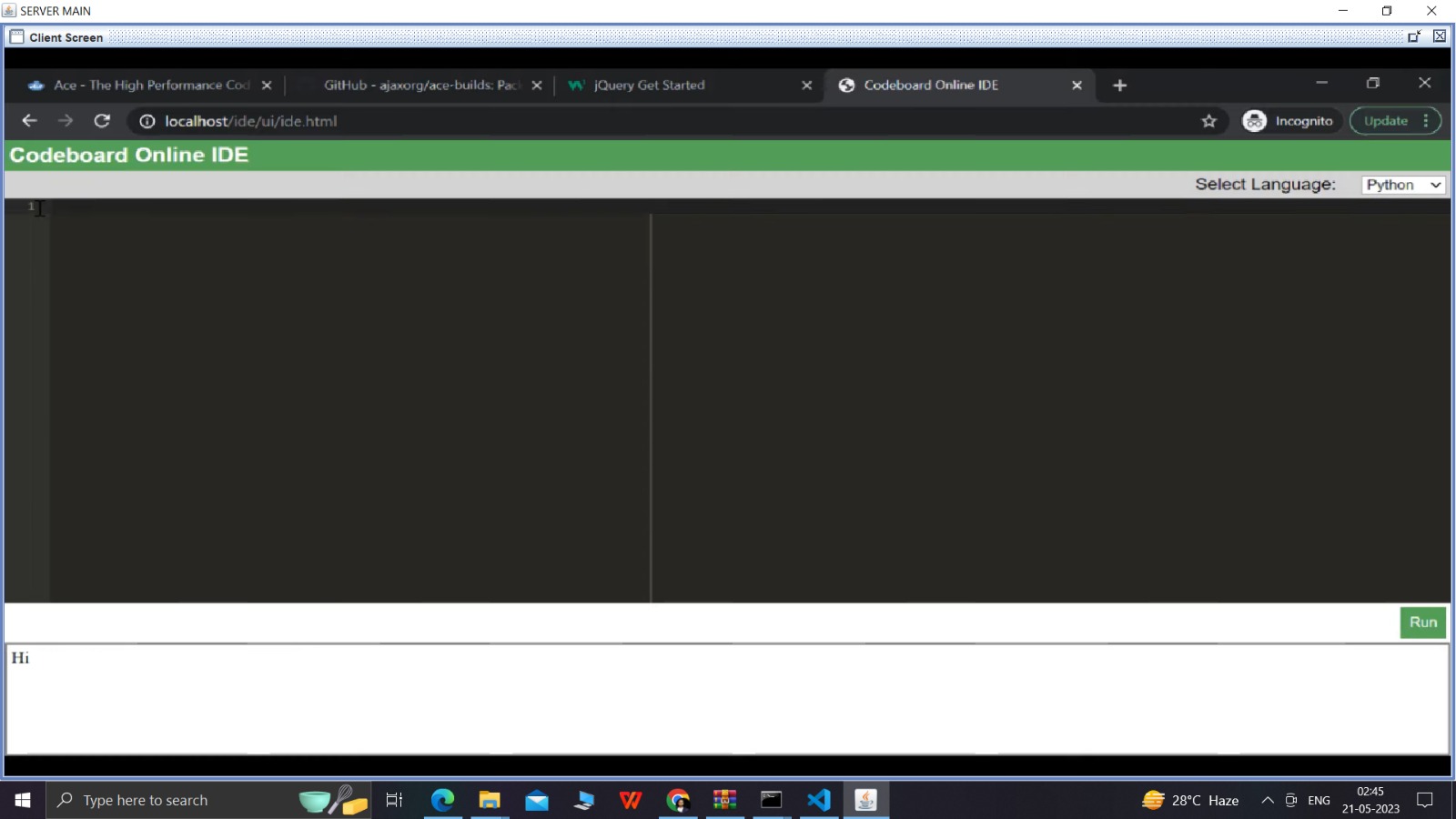
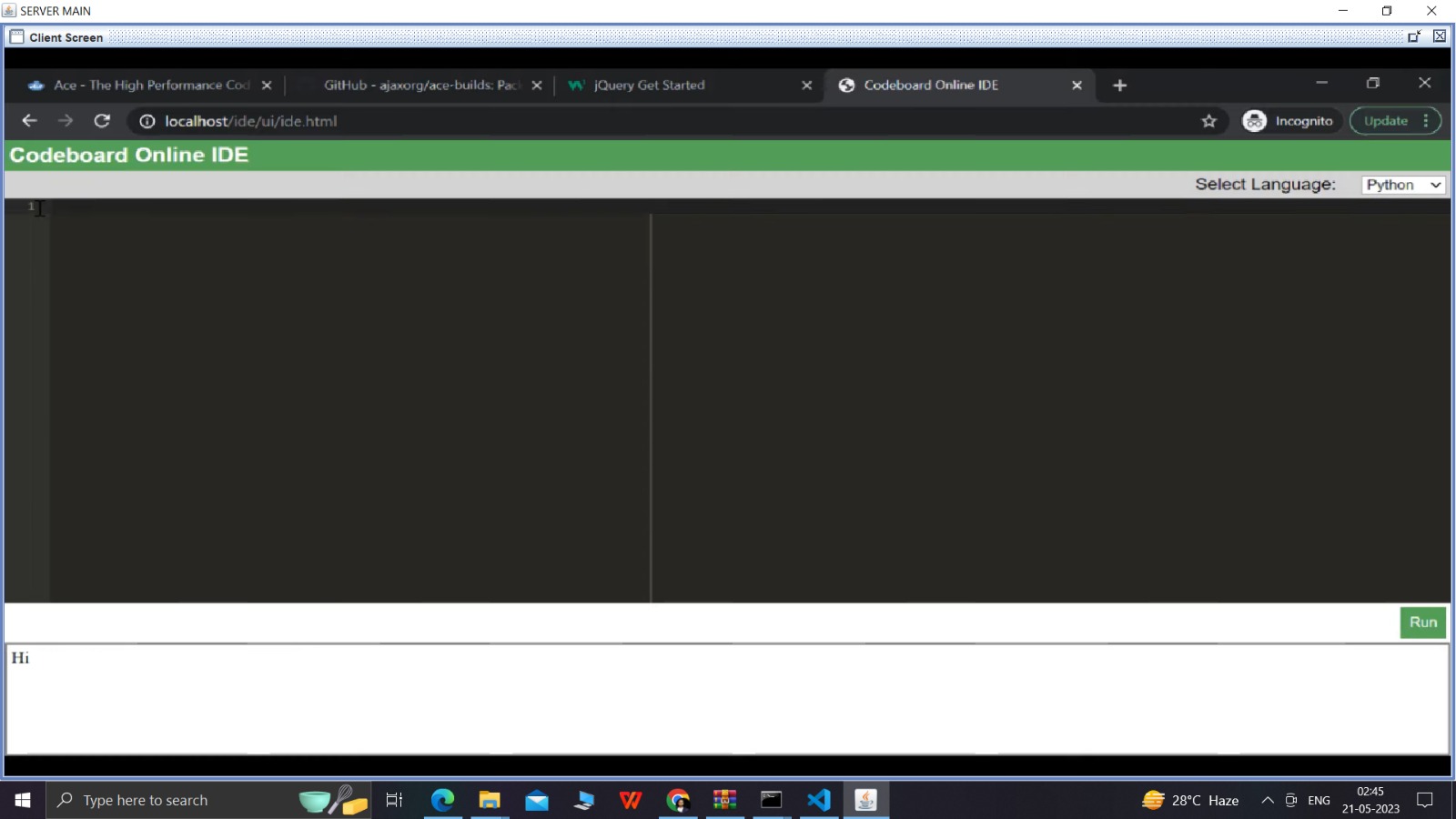
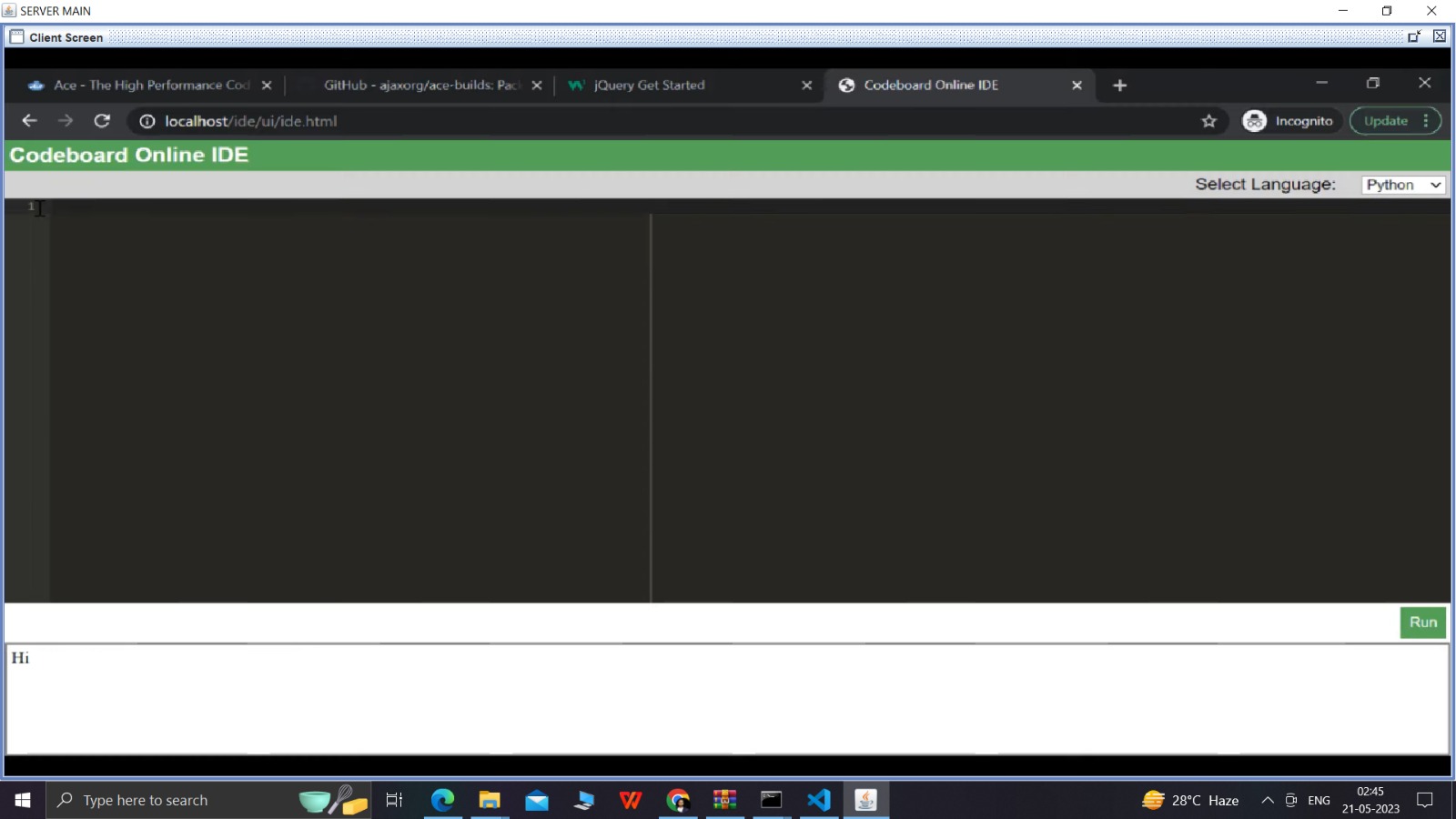
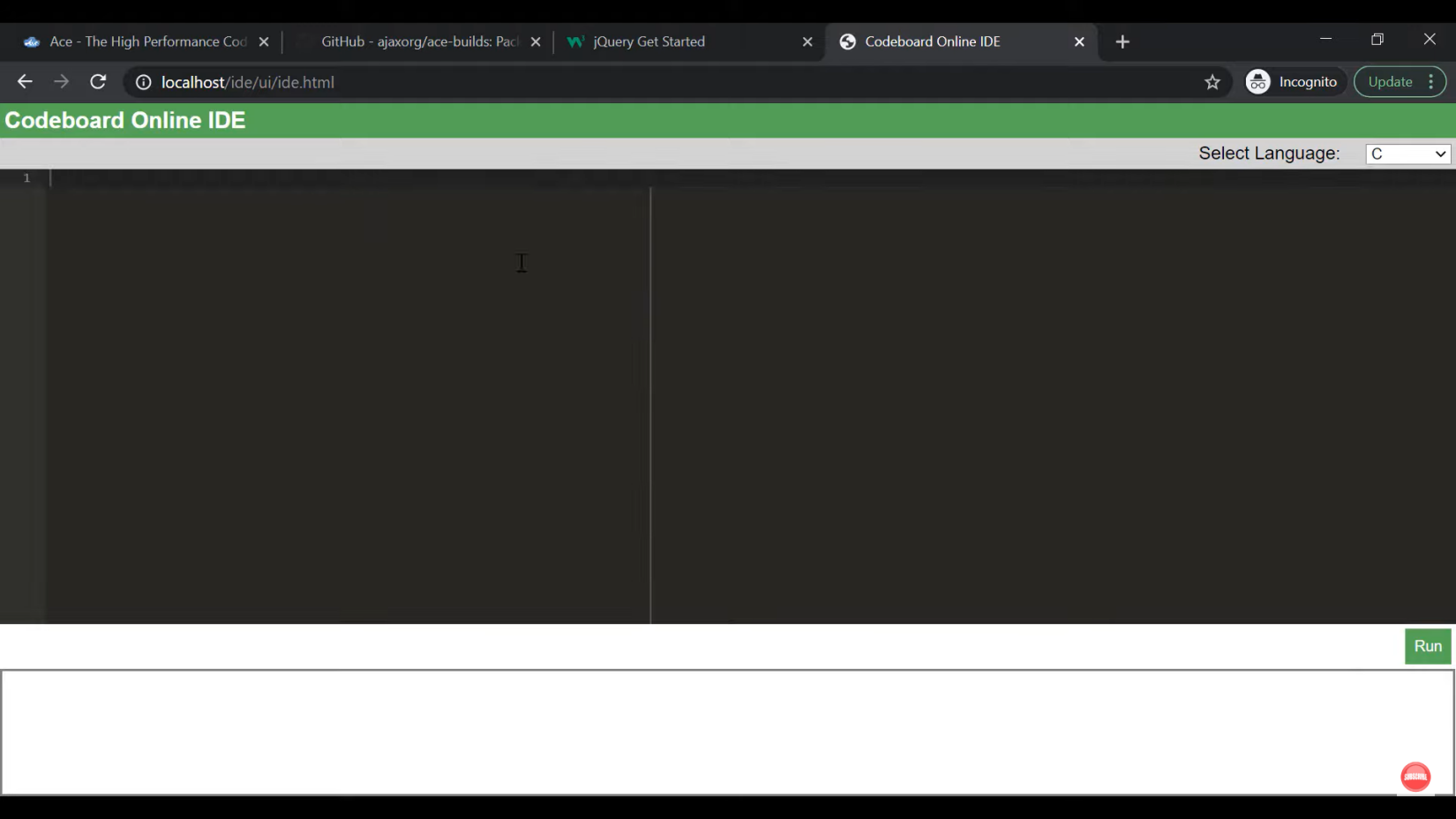


Fig. Client-Server Architecture

1. **How to use ?**

* Connect two or more laptops or PCs in same network.
* Start server program on one PC or laptops. Depending on the setup of your remote server, you may need to use a terminal or SSH client to log in. Once you're connected, you should be able to access the IDE remotely. Use port as 5000.
* Start client program on other PCs or laptop. Enter LAN IP of server PC (obtained by ipconfig/ifconfig on server PC for Windows/Linux) and port as 5000.
* Once connected properly, you can view and control client PC from server PC.
* Start Coding : Once everything is configured, you should be able to start developing code as if you were working locally. Any changes you make will be saved to the remote server, and you'll be able to run and debug your code from the IDE.

1. **SNAPSHOT**



1. **Conclusion and Future Scope**

In conclusion, our team has successfully developed a remote IDE that enables developers to code and collaborate from anywhere in the world. Our platform offers a rich set of features, including real-time collaboration, version control, and debugging tools, which make it easy for teams to work together seamlessly.

Throughout the project, we faced several challenges, such as ensuring secure access to the system and optimizing performance across different devices and networks. However, we overcame these challenges by conducting thorough testing and implementing robust security protocols.

Moving forward, we recommend further improvements to enhance user experience and streamline the development process. For instance, we could explore integrating more third-party tools and services or developing mobile applications to expand our reach and accessibility.

We would like to express our gratitude to the stakeholders who supported this project and the team members who contributed their time and expertise to make it a success. We believe that our remote IDE will have a positive impact on the development community and look forward to seeing it used and adopted widely**.**

**8. Bibliography**

1. Stack Over flow
2. Github
3. JavaTPoint
4. Geeks for Geeks
5. BAI Chat