CS 520: Software Engineering
Team MSC
Christopher Gomez, Ran He, Mohini Jain, Venkata Samyukta Malapaka
Professor Heather Conboy
05/22/2023

Comparing features and performances between the Visual Studio Code and IntelliJ IDEA Integrated Development Environments.

Introduction

Developers often encounter significant challenges throughout their pursuit of determining the most appropriate integrated development environments (IDEs). This problem is only further complicated by the fact that many best-IDE choices differ based on the individual's requirements, constraints, and programming needs. The main objective for our project is to evaluate Visual Studio code and IntelliJ, two of the most popular Java IDEs, through comparing and contrasting the features and performances of both IDEs. Both IDEs are well-known within the Java community and come with their own share of distinctive features, advantages, disadvantages, and limitations. The primary aim of this study is to develop a holistic view of both IDEs, through weighing their comparative pros and cons, thus allowing readers to more easily pick the development environment of their choice depending on their specific use constraints and development requirements. This brings us to our main research/evaluation question: how do the features, advantages, disadvantages, and limitations of Visual Studio Code and IntelliJ IDEA compare, and what is the most appropriate IDE for developers based on their constraints and requirements? We will evaluate this question through testing both IDEs along a variety of different metrics such as - syntax highlighting, code completion, debugging tools, and version control integration.

Approach

The first step in our approach is to conduct a comprehensive evaluation of the two IDEs. Using the criteria that we have defined. For example, we might look at the level of support available for each IDE, including things like online documentation, user forums, and technical support services. We might also consider the overall cost of using each IDE, including any licensing fees or subscription costs. In addition, we will also consider other factors like functionalities, popularity with the public and overall user experience of each IDE.

Once we have completed the evaluation process, we will develop a test code that will cover the major features and functionalities of both IDEs. This test code will be designed to measure the performance of each IDE in terms of execution speed, memory usage, Version

control.We will run this test code on both IDEs and compare the results to determine which IDE performs better in terms of performance and efficiency. We will also consider what we evaluated during the first step of our approach.

Overall, our approach is designed to provide a comparison of the two IDEs based on a set of predefined criteria and performance testing.

Methodology - tools, evaluation criteria

The tools we'll be comparing throughout this study are the VSCode IDE and IntelliJ IDEA. We'll be comparing these two tools using a wide variety of evaluation criteria meant to encapsulate comparing the two development environments. The evaluation criteria are unit testing, debuggability, git integration and version control, price, installation process, OS compatibility, download size, understandability, usability (user interaction), color theme, language independency, maintainability/stability, code autocompletion.

Both VSCode and IntelliJ support unit testing through their own frameworks, thus for evaluation we can compare both the features and capabilities of their respective unit testing frameworks. Similarly, both IDEs have robust debugging tools, so we can compare their ease of use and functionality through their respective debugging features such as breakpoints and inspecting variables. Since both VSCode and IntelliJ support git integration, we can compare the two based on the ease of cloning/setting up a repository, branch management, and other git features. For price, VSCode is free and open-source whereas IntelliJ IDEA has a free community edition and a paid ultimate edition, so we can simply compare the two using the cost difference as a metric. For the installation process, we can evaluate based on the ease of installing the IDEs for the operating systems we have access to, namely Windows and Mac. For downloading we can simply use the download size as the evaluation metric and compare the two IDEs based solely on their download size. For usability we can compare the ease of use and learnability of each IDE through ease of navigating the IDE and documentation. For understandability we can compare the two IDEs based on the availability of their documentation, tutorials, and other support materials. For color themes, we can evaluate the two IDEs based on the number of preset color themes they have available. For language independency we can evaluate the IDEs based on the number of programming languages they're able to support. For maintainability/stability we can look into the frequency and number of patches for the software. Lastly, for code autocompletion we can evaluate the IDEs based on how well each IDE suggests and completes code snippets as the user types, especially in regards to the suggested code's accuracy and relevance.

Evaluation Criteria

- **Price:**

In this section, we compare the tools based on their pricing models and essentially evaluate them on their costs and benefits as per the options provided.

Visual Studio Code: While Visual Studio Code is free to download and use, some extensions and add-ons may require payment. Additionally, developers may choose to purchase a Visual Studio subscription, which provides access to additional features and tools, such as cloud services, advanced debugging capabilities, and more. However, the base version of Visual Studio Code is entirely free.

IntelliJ IDEA:
The IDE operates on a subscription-based pricing model. The subscription pricing varies based on the number of users and the duration of the subscription. JetBrains offers monthly, yearly, and perpetual licenses. The perpetual license allows the user to use the current version of the software indefinitely, but does not include access to new versions released after the subscription expires. This is primarily divided into three categories:
a) For Organizations -
It has two options for download:
For the first option, IntelliJ IDEA Ultimate, the pricing per user is listed as follows -
- First year - $599.00
- Second year - $479.00
- Third year onwards - $359.00
And for the second option, All Products Pack (includes three extensions, two profilers, and a collaborative development service), the pricing per user is as listed follows -
- First year - $779.00
- Second year - $623.00
- Third year onwards - $467.00

b) For Individuals -
It has two options for download. For the first option, IntelliJ IDEA Ultimate, the pricing per user is listed as follows -
- First year - $169.00
- Second year - $135.00
- Third year onwards - $101.00
And for the second option, All Products Pack (includes three extensions, two profilers,

and a collaborative development service), the pricing per user is listed as follows -
- First year - $289.00
- Second year - $231.00
- Third year onwards - $173.00

- Having said that, IntelliJ IDE also provides a free of cost, open source version named Community Edition (licensed under Apache 2.0) that provides some basic features for JVM and Android development.
- A thirty days free trial is provided for the commercially distributed version, IntelliJ IDEA Ultimate that has additional features.

c) Special Offers:
IntelliJ IDEA provides various special offers and promotions with cost benefits on their subscriptions. These offers include academic pricing for students and faculty, discounts for early-stage startups through the Startup Program, and volume discounts for organizations that purchase multiple licenses.
One of the primary special offers provided by IntelliJ IDEA is its academic pricing that allows students and educators to access the Ultimate Edition of IntelliJ IDEA for educational purposes at no cost.

● System Requirements for IDE Installation:

There are four subsections in which this evaluation criterion can be divided. These majorly focus on the minimum hardware and software expectations at which the IDE can be installed and run glitch free on users' systems.

- **Operating System Compatibility:**

  Visual Studio Code: VS Code is supported on the following platforms:
- Windows 8.0, 8.1 and 10, 11 (32-bit and 64-bit)
- OS X High Sierra (10.13+)
- Linux (Debian): Ubuntu Desktop 16.04, Debian 9
- Linux (Red Hat): Red Hat Enterprise Linux 7, CentOS 7, Fedora 34

  The official Visual Studio Code website recommends using a Linux distribution that includes GLIBC 2.27 or later. However, it is possible to run Visual Studio Code on Linux distributions that do not include GLIBC 2.27 or later by manually installing the required version of GLIBC.

  IntelliJ IDEA: The IDE is compatible with 64-bit versions of the following:

- Microsoft Windows 10 1809 or later
- Windows Server 2019 or later
- macOS 10.15 or later
- Any Linux distribution that supports Gnome, KDE, or Unity DE.

IntelliJ IDEA is not available for the Linux distributions that do not include GLIBC 2.27 or later.

- **Hardware Requirements:**

  Visual Studio Code: Any modern CPU with 1.6 GHz or higher processor and a minimum of 1 GB RAM will be able to smoothly run VSCode provided all the other dependencies are satisfied.

  IntelliJ IDEA: Any modern CPU with 2 GB of free RAM and * GB of total RAM can efficiently run Intellij IDEA in a system.

- **Disk Space:**
  IntelliJ IDEA is a feature-rich, full-fledged integrated development environment (IDE) that provides developers with a wide range of tools and features to support various programming languages and development workflows. As a result, it may require more disk space to accommodate the larger number of features and functionalities that it offers.
  On the other hand, Visual Studio Code is a lightweight, open-source code editor that focuses on providing a fast and streamlined development experience. While it also provides many features and extensions, it is designed to be modular, allowing developers to choose only the features and extensions they need, which may result in a smaller disk space requirement.
  Overall, the difference in disk space requirements between IntelliJ IDEA and Visual Studio Code is primarily due to the design and feature set of each IDE. IntelliJ IDEA's comprehensive feature set and built-in language support require more disk space, while Visual Studio Code's lightweight and modular design allows it to offer a smaller disk space requirement.

  Visual Studio Code: VSCode in this criterion comes out as extremely lightweight with a file download size of less than 200 MB and disk footprint of less than 500 MB.

  IntelliJ IDEA: The basic version of the platform requires a minimum of 3.5 GB of disk space.

- **Other Dependencies:** In terms of Java installation, since VSCode is largely a code editor, it does not require installation of Java since it uses language servers and extensions that provide language specific features that support different programming languages.
  IntelliJ IDEA on the other hand, provides a complete development environment which requires prior installation of Java or Python as per the developers' requirements.

• Version control:

In this section the scope of version control within the two IDEs is analyzed. Provision of built-in version control systems is crucial in ensuring efficient and error free project development when multiple users are involved. Not only does it enhance the collaboration amongst the developers involved, it also largely eliminates the errors of code overriding and other miscommunications in terms of code development. IntelliJ IDEA supports several version control systems, including Git, Subversion, Mercurial, and CVS. It provides a rich set of features for working with version control systems, such as support for branching and merging, code review tools, and integration with popular hosting services such as GitHub and Bitbucket. IntelliJ IDEA also includes a powerful diff tool that allows developers to compare code changes and identify differences between files.
Similarly, VSCode also has built-in support for Git and provides many of the same features as IntelliJ IDEA, such as branching and merging, code review tools, and diff tools. In addition, VSCode includes support for other version control systems such as Subversion and Perforce through extensions.

Visual Studio Code:

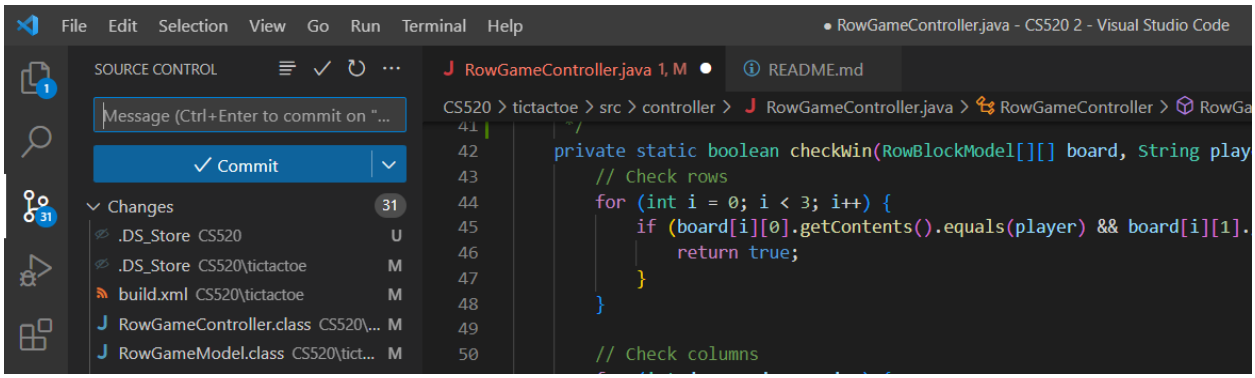


Fig: VSCode Version Control Using Git

IntelliJ IDEA:

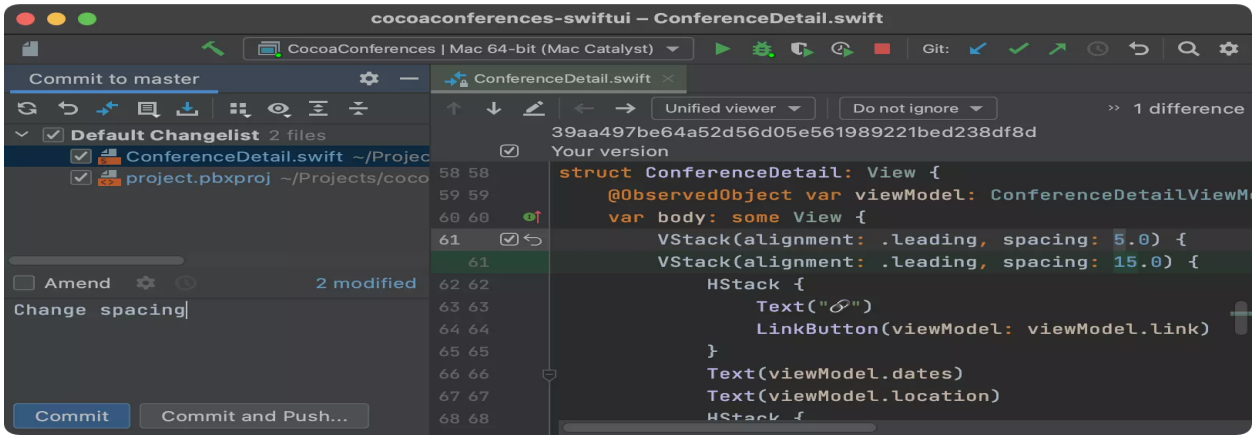


Fig: IntelliJ IDEA Version Control Using Git

- ## Color theme / aesthetics

Visual Studio Code:
There are several out-of-the-box color themes in VS Code for one to try.

Many themes have been uploaded to the VS Code Extension Marketplace by the community. The process of changing the color theme is to pick a new theme, install it and restart VS Code and the new theme will be available.

You can search for themes in the Extensions view (⇧⌘X) search box using the @category:"themes"filter.

Windows and macOS support light and dark color schemes. There is a setting, window.autoDetectColorScheme, that instructs VS Code to listen to changes to the OS's color scheme and switch to a matching theme accordingly.

To customize the themes that are used when a color scheme changes, you can set the preferred light, dark, and high contrast themes with the settings:

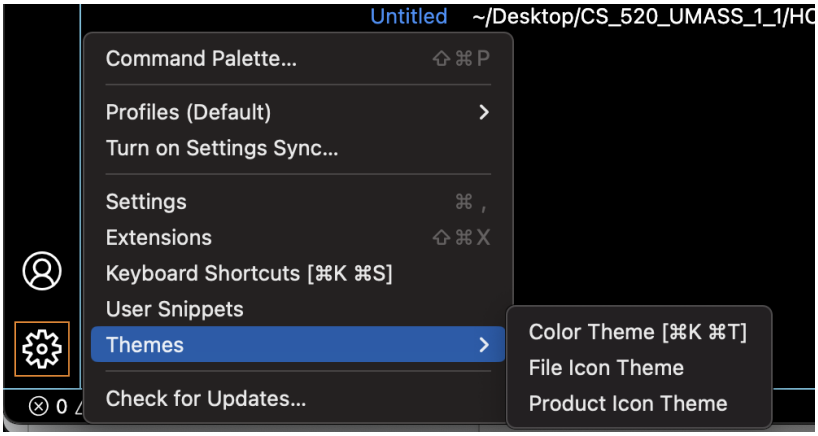


Fig: Visual Studio Code Theme change

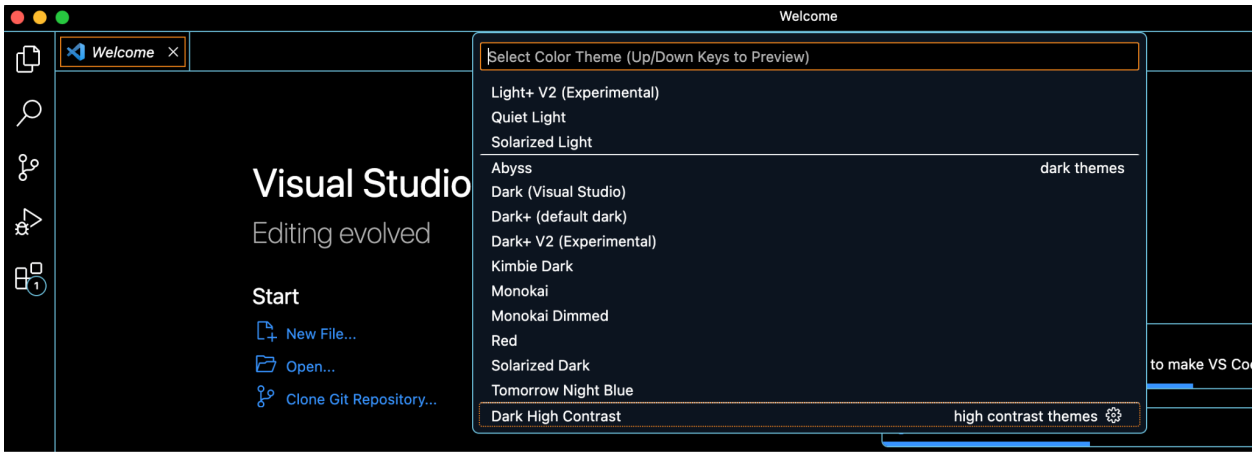


Fig: Visual Studio Code in-built themes

IntelliJ IDEA:

IntelliJ IDEA lets you choose between configurable color schemes that define colors and fonts used in IDE text. To define color and font settings, open the Editor | Color Scheme page of the IDE settings ⌘ Comma.

Steps to select a color scheme:

Press ⌘ Comma to open the IDE settings and select Editor | Color Scheme. Use the Scheme list to select a color scheme.

By default, there are the following predefined color schemes:

Classic Light: designed for the macOS Light and Windows 10 Light interface themes

Darcula: designed for the Darcula interface theme

High contrast: designed for the High contrast interface theme (recommended for users with sight deficiency)

IntelliJ Light: designed for the IntelliJ Light interface theme

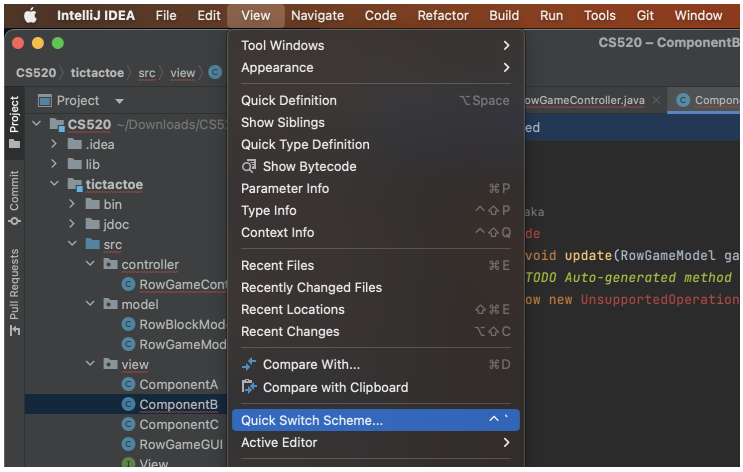


Fig: IntelliJ IDEA theme

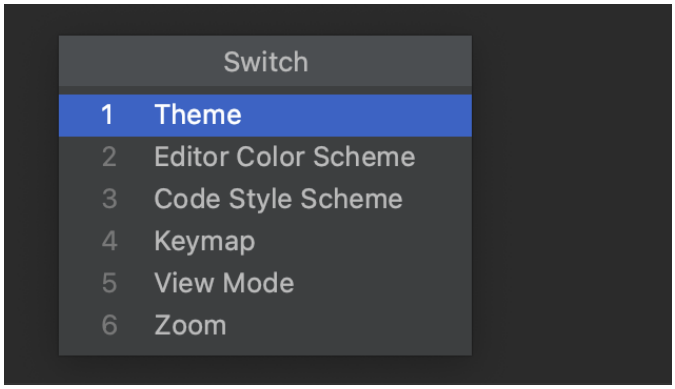


Fig: IntelliJ IDEA in-built themes

- ## Language independency

IntelliJ IDEA and Visual Studio Code (VSCode) are widely used integrated development environments (IDEs) that cater to the needs of developers across multiple programming languages. When considering language independence, there are distinct characteristics and considerations for each IDE:

IntelliJ IDEA:

Strengths: IntelliJ IDEA is renowned for its extensive language support and deep integration with various programming languages. It offers comprehensive tools and features specifically designed for languages such as Java, Kotlin, Scala, Python, and more. These features encompass advanced code analysis, debugging capabilities, refactoring tools, and productivity-enhancing functionalities that are tailored to specific languages.
Considerations: While IntelliJ IDEA excels in providing comprehensive language support, it may require a relatively higher learning curve compared to VSCode. It is a full-fledged IDE with a rich feature set, which can result in increased memory usage and slower startup times.

Visual Studio Code (VSCode):

Strengths: VSCode emphasizes extensibility and supports a broad range of programming languages through its marketplace extensions. It offers a lightweight and flexible development environment that accommodates multiple languages. The marketplace hosts numerous language-specific extensions, enabling features like syntax highlighting, IntelliSense, and debugging capabilities.
Considerations: Although VSCode boasts an extensive extension ecosystem, the depth of language-specific features might not match that of IntelliJ IDEA. Advanced features and language-specific tools available in IntelliJ IDEA may not be as abundant or powerful in VSCode.

In conclusion, both IntelliJ IDEA and VSCode can support various programming languages, but IntelliJ IDEA stands out for its specialized and comprehensive language-specific features. On the other hand, VSCode provides a lightweight and adaptable development environment with extensive extension support. The choice between the two depends on the specific programming languages utilized, personal workflow preferences, and the depth of language-specific tooling required.

- ## Maintainability / Stability

When comparing the maintainability and stability of IntelliJ IDEA and Visual Studio Code (VSCode), several factors come into play. Here's an analysis of both IDEs in terms of maintainability and stability:

IntelliJ IDEA:

Maintainability: IntelliJ IDEA is known for its robust and reliable nature. It is developed by JetBrains, a company with a strong track record of producing high-quality developer tools. JetBrains provides regular updates and bug fixes to ensure the IDE remains stable and up to date. Additionally, IntelliJ IDEA offers a rich set of features for refactoring, code analysis, and project management, which contribute to code maintainability.
Stability: IntelliJ IDEA is generally considered a stable IDE. It undergoes rigorous testing before each release to minimize the presence of bugs and stability issues. However, being a complex IDE with numerous features, occasional stability issues can occur, especially when working with large codebases or in combination with certain plugins or configurations. JetBrains actively addresses reported issues and releases patches and updates to improve stability.

Visual Studio Code (VSCode):

Maintainability: VSCode benefits from being an open-source project supported by Microsoft and a vast community of developers. This enables frequent updates, bug fixes, and continuous improvements to enhance maintainability. Additionally, the extensible nature of VSCode allows developers to create and contribute to various extensions, providing additional functionality and support for different languages.
Stability: VSCode generally maintains a good level of stability. Being a lightweight IDE, it offers a less complex environment compared to IntelliJ IDEA, which often results in fewer stability issues. However, the stability can still be influenced by the extensions and configurations used. Some extensions may be less stable or not actively maintained, potentially introducing stability concerns. Microsoft actively addresses reported issues and releases updates to enhance stability.
Overall, both IntelliJ IDEA and VSCode strive to provide maintainable and stable development environments. IntelliJ IDEA's robust development by JetBrains and its feature-rich nature contribute to its strong maintainability, while VSCode's open-source nature and active community support contribute to its maintainability. Stability can vary depending on factors such as the complexity of the IDE, the usage of extensions, and the specific configurations employed in each case. It's advisable to keep the IDE and its

extensions up to date to ensure the best possible stability and to report any issues encountered to the respective development teams.

- ## Intelligent code completion

When it comes to intelligent code completion, both IntelliJ IDEA and Visual Studio Code (VSCode) offer powerful features to assist developers in writing code more efficiently. Here's a comparison of intelligent code completion in IntelliJ IDEA and VSCode:

IntelliJ IDEA:

Intelligent Code Completion: IntelliJ IDEA is renowned for its advanced and context-aware code completion capabilities. It analyzes the codebase, understands the context, and suggests relevant code completions, including classes, methods, variables, and even specific code patterns. It takes into account the current scope, imported packages, and inferred types to provide accurate and contextually appropriate suggestions.
Smart Context Assistance: IntelliJ IDEA goes beyond basic code completion. It offers smart context assistance, such as parameter hints, quick fixes, and intention actions. These features suggest and apply changes to code based on potential improvements or detected issues, streamlining the development process and reducing manual effort.

Fig: IntelliJ IDEA intelligent code completion

Visual Studio Code (VSCode):

IntelliSense: VSCode incorporates IntelliSense, a powerful code completion system that provides suggestions for variables, functions, classes, and more. It leverages static analysis, type inference, and language service extensions to offer contextually relevant completions. IntelliSense in VSCode is highly customizable and can be extended with language-specific extensions available in the marketplace.
Snippet Support: VSCode includes support for code snippets, allowing developers to define custom code templates that can be expanded with a few keystrokes. Snippets can include placeholders for variables and provide pre-defined code structures, making it easier to write repetitive code patterns.

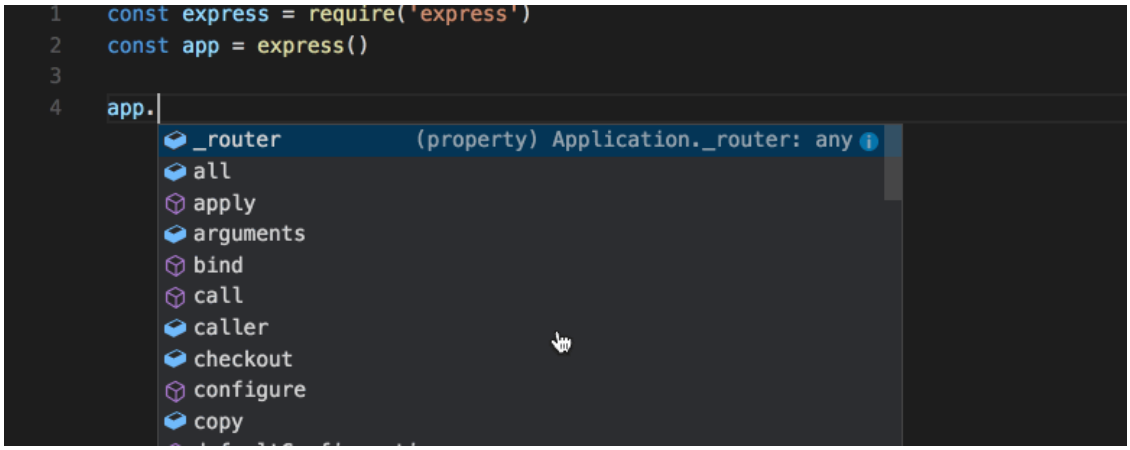


Fig: VS Code intelligent code completion

- ## Support and Documentation provided

IntelliJ IDEA and Visual Studio Code (VSCode) both offer support and documentation to assist developers in using their respective IDEs effectively. Here's a comparison of the support and documentation provided by IntelliJ IDEA and VSCode:

IntelliJ IDEA:

Official Documentation: JetBrains, the company behind IntelliJ IDEA, provides extensive and well-documented resources on their official website. The documentation covers various topics, including installation, setup, features, tutorials, and specific language integrations. It includes detailed explanations, code examples, and step-by-step guides to help users navigate the IDE's features and functionality.

Community and Forums: IntelliJ IDEA has an active community of users, and JetBrains maintains an official community forum where users can seek help, ask questions, and engage in discussions. JetBrains developers and community members often provide assistance and share insights to address issues and provide guidance.
JetBrains Support: JetBrains offers different support options, including paid support plans for IntelliJ IDEA. These plans provide direct access to JetBrains' technical support team, ensuring prompt assistance for more complex issues or inquiries.
Visual Studio Code (VSCode):

Official Documentation: Microsoft provides comprehensive documentation for VSCode on their official website. The documentation covers various aspects of using the IDE, including installation, configuration, features, debugging, extensions, and language-specific integrations. It includes detailed explanations, code samples, and tutorials to help users make the most of VSCode's capabilities.
Community and Forums: VSCode has a large and vibrant community of developers. The official VSCode community forum allows users to ask questions, share knowledge, and seek guidance from the community. Microsoft developers and community members actively participate in discussions and provide support.
GitHub Repository: VSCode is an open-source project hosted on GitHub. The GitHub repository serves as a valuable resource, providing access to the source code, issue tracker, and community-contributed extensions. Users can find additional support and documentation through community-driven repositories, samples, and discussions.

- ## Source and Version Control

IntelliJ IDEA and Visual Studio Code (VSCode) both offer support and documentation to assist developers in using their respective IDEs effectively. Here's a comparison of the support and documentation provided by IntelliJ IDEA and VSCode:

IntelliJ IDEA:

- IntelliJ IDEA provides robust support for source control systems, including Git, Subversion (SVN), Mercurial, and others.
- Developers can initialize a new repository, clone an existing repository, or open a project that is already under version control.
- The Version Control tool window in IntelliJ IDEA displays the current status of files in the repository, including changes, additions, deletions, and untracked files.
- Developers can easily commit changes to the repository directly from within the IDE, along with providing commit messages and selecting files to include.

- Branches can be created, switched, merged, and deleted seamlessly within IntelliJ IDEA.
- The IDE offers a visual diff tool that highlights the differences between file versions, making it easy to review changes before committing.
- IntelliJ IDEA supports advanced Git operations, such as interactive rebase, cherry-picking, and resolving merge conflicts.
- Developers can view and manage commit history, including the ability to browse previous revisions and compare changes.
- IntelliJ IDEA integrates with popular code hosting platforms like GitHub, Bitbucket, and GitLab, providing features such as pull requests, code reviews, and direct integration with issue tracking systems.
- The IDE offers tools for resolving merge conflicts, including a merge conflict editor and three-way merge capabilities.
- Annotated views and blame annotations allow developers to see who last modified each line of code and view commit details.
- IntelliJ IDEA provides keyboard shortcuts, context menus, and toolbar actions for efficient source control operations.
- The IDE supports configuring and managing multiple repositories simultaneously, allowing developers to work with different projects and repositories seamlessly.
- IntelliJ IDEA's source control features provide a smooth and intuitive workflow for managing version control operations, collaborating with team members, and maintaining code quality throughout the development process.

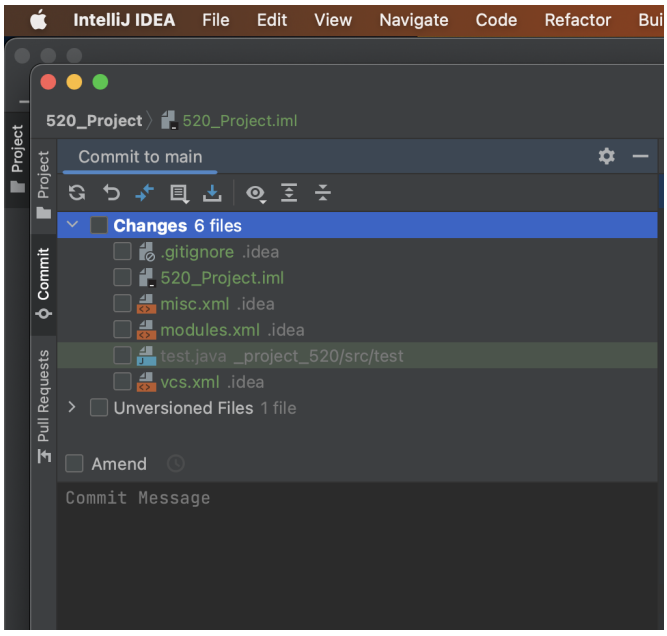


Fig: IntelliJ IDEA source control

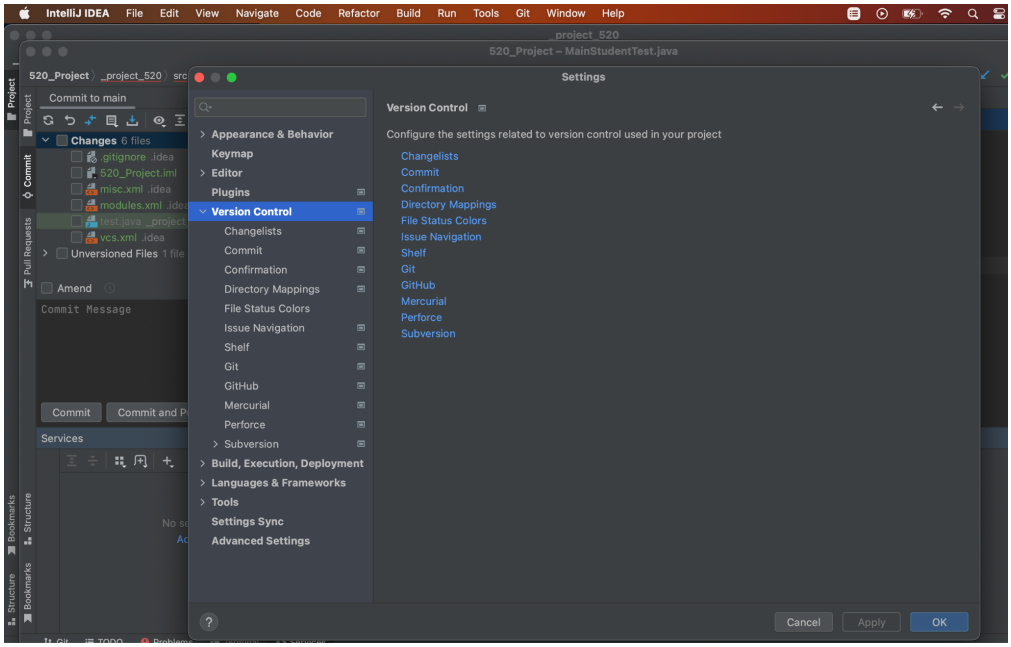


Fig: IntelliJ IDEA version control

Visual Studio Code (VSCode):

- Visual Studio Code provides built-in support for popular version control systems such as Git, Mercurial, and SVN.
- Developers can initialize a repository, clone an existing repository, or open a folder that is already under version control.
- The Source Control view in VS Code displays the current repository status, including modified, added, deleted, and untracked files.
- Developers can stage changes and commit them to the repository directly from the Source Control view.
- Branches can be created, switched, and merged seamlessly within the IDE.
- Visual Studio Code offers powerful visual diff and merge tools to compare changes between files or branches.
- Interactive rebasing, cherry-picking, and other advanced Git operations can be performed within the IDE.
- VS Code integrates with popular Git services like GitHub, GitLab, and Bitbucket, providing features such as pull requests and code reviews.
- Extensions and plugins are available to enhance the source control experience in Visual Studio Code, offering additional functionality and integrations with other tools.

- Keyboard shortcuts and commands are provided to perform common source control operations quickly and efficiently.
- Visual Studio Code's source control features enable developers to effectively manage their code versions, collaborate with others, track changes, and maintain code quality throughout the development process.

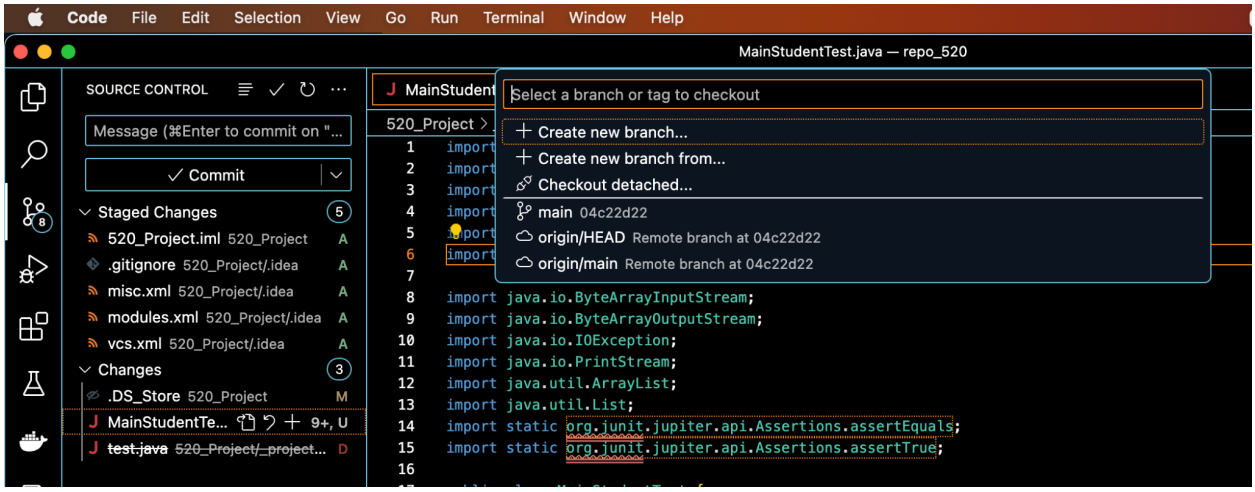


Fig: VS Code source control

- ## Integrated Debugger

IntelliJ IDEA:

- IntelliJ IDEA offers an integrated debugger that enables developers to debug their code directly within the IDE.
- The debugger supports multiple programming languages, including Java, Kotlin, Scala, and more.
- Developers can set breakpoints in their code to pause execution at specific lines or conditions.
- The debugger provides a range of debugging commands, such as stepping over, stepping into, and stepping out of code execution.
- The Variables view in IntelliJ IDEA allows developers to inspect and modify variables' values during debugging.
- The Call Stack view displays the current call stack, enabling developers to navigate through function calls and frames.
- Conditional breakpoints can be set to pause execution only when specific conditions are met.

- IntelliJ IDEA's debugger integrates with external tools and frameworks, such as JUnit for unit testing and application servers for remote debugging.
- The Watches view enables developers to monitor and evaluate expressions and variables continuously during debugging.
- The IDE offers advanced debugging features, including evaluate expression, evaluate code fragment, and exception breakpoints.
- IntelliJ IDEA's debugger provides comprehensive support for debugging multi-threaded applications and concurrent code.
- Debugging configurations can be customized and saved for easy reuse in different scenarios or projects.
- The IDE offers keyboard shortcuts and context-aware suggestions for efficient debugging workflows.
- IntelliJ IDEA's integrated debugger enhances developer productivity by providing a seamless and powerful debugging experience within the IDE environment.

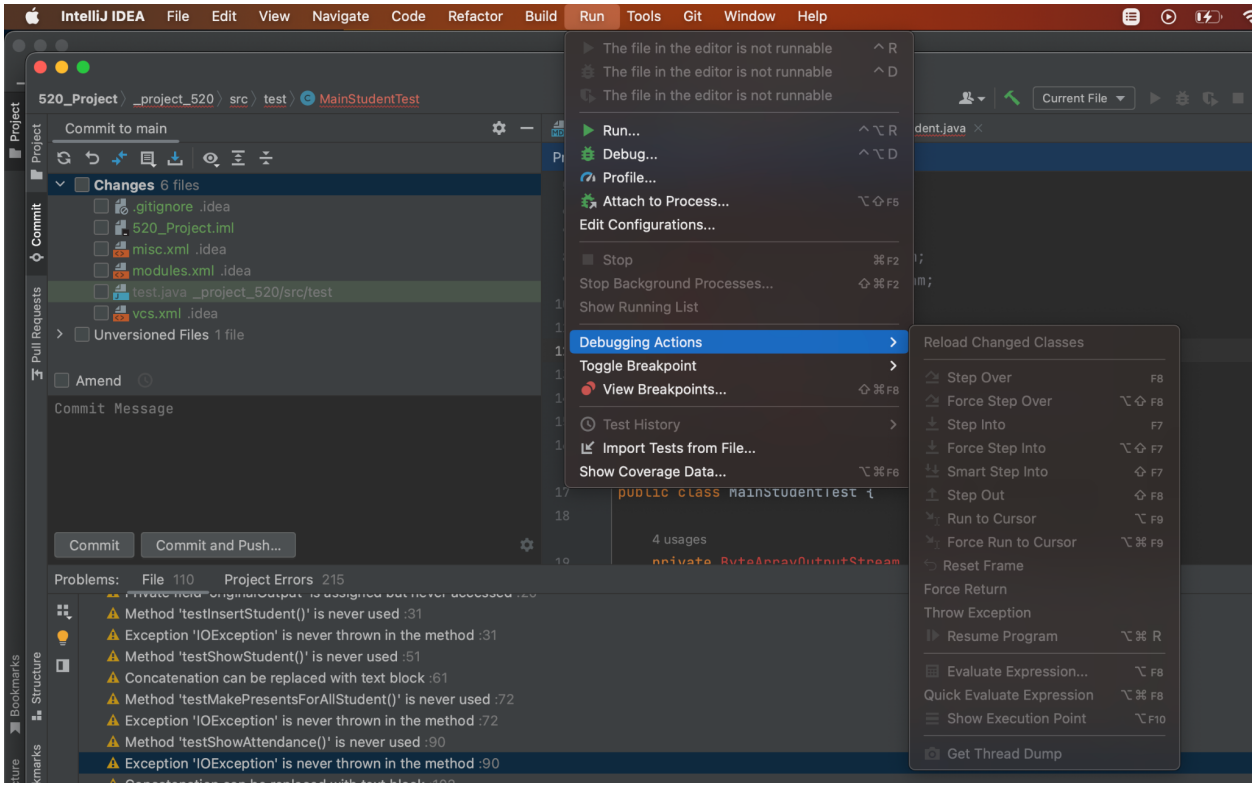


Fig: IntelliJ IDEA debugging

Visual Studio Code (VSCode):

- Visual Studio Code provides an integrated debugger that allows developers to debug their code directly within the IDE.
- The debugger supports multiple programming languages, including JavaScript, Python, C++, and more.
- Developers can set breakpoints in their code to pause execution at specific lines or conditions.
- The Variables view in VS Code allows developers to inspect and modify variables' values during debugging.
- The Call Stack view displays the current call stack, allowing developers to navigate through function calls and frames.
- VS Code offers step-by-step debugging features, including stepping over, stepping into, and stepping out of code execution.
- Conditional breakpoints can be set to pause execution only when specific conditions are met.
- The Watch view enables developers to monitor and evaluate expressions and variables continuously during debugging.
- Visual Studio Code's debugger integrates with external tools and services, such as Chrome DevTools for web debugging or remote debugging on servers.
- Debugging configurations can be customized and saved for easy reuse in different scenarios or projects.
- VS Code provides a rich set of debugging features, including exception handling, console output, debug console, and more.
- The IDE offers keyboard shortcuts and command palette options for efficient debugging workflows.
- Visual Studio Code's integrated debugger enhances developer productivity by providing a seamless debugging experience within the IDE environment.

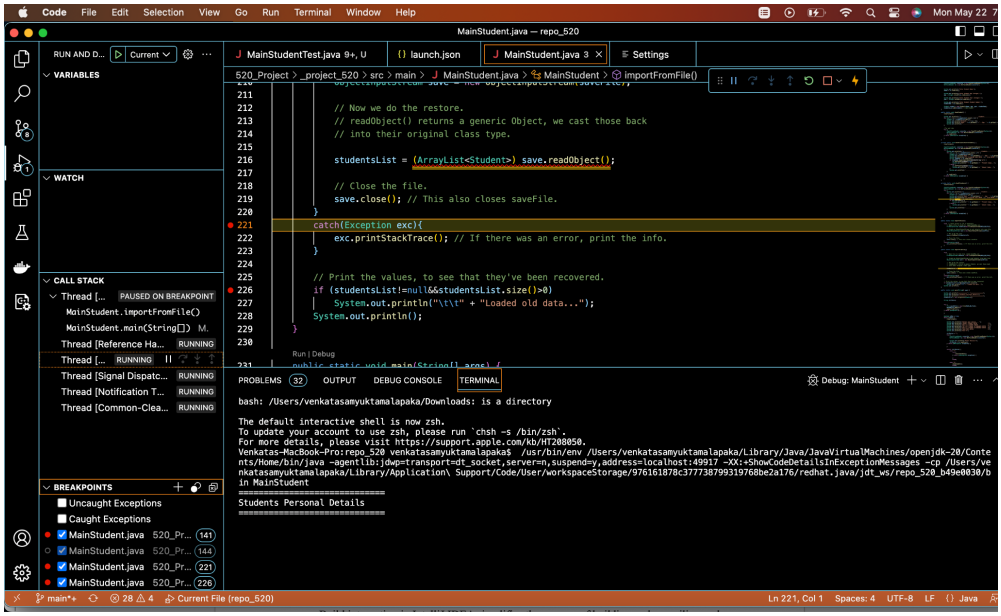


Fig: VS Code source control

- **Build Automation**

  Build automation in an IDE refers to the ability of an Integrated Development Environment (IDE) to automate the process of building, compiling, and running your code. IDEs often provide built-in tools or integrations with build systems to streamline these tasks.

  IntelliJ IDEA:

  Build integration in IntelliJ IDEA simplifies the process of building and compiling code directly within the IDE. It provides a seamless workflow for executing build tasks and viewing build output without the need to rely on external build tools or command-line interfaces. With IntelliJ IDEA's build integration, developers can efficiently automate their build processes, streamline their development workflow, and conveniently access build-related features and information within the IDE environment.

1. Build integration in IntelliJ IDEA allows developers to automate the build process directly within the IDE.
2. It supports various build systems and technologies such as Maven, Gradle, Ant, and more.
3. Developers can configure build tasks and customize them based on project requirements.
4. The IDE provides a user-friendly interface for managing build configurations and dependencies.

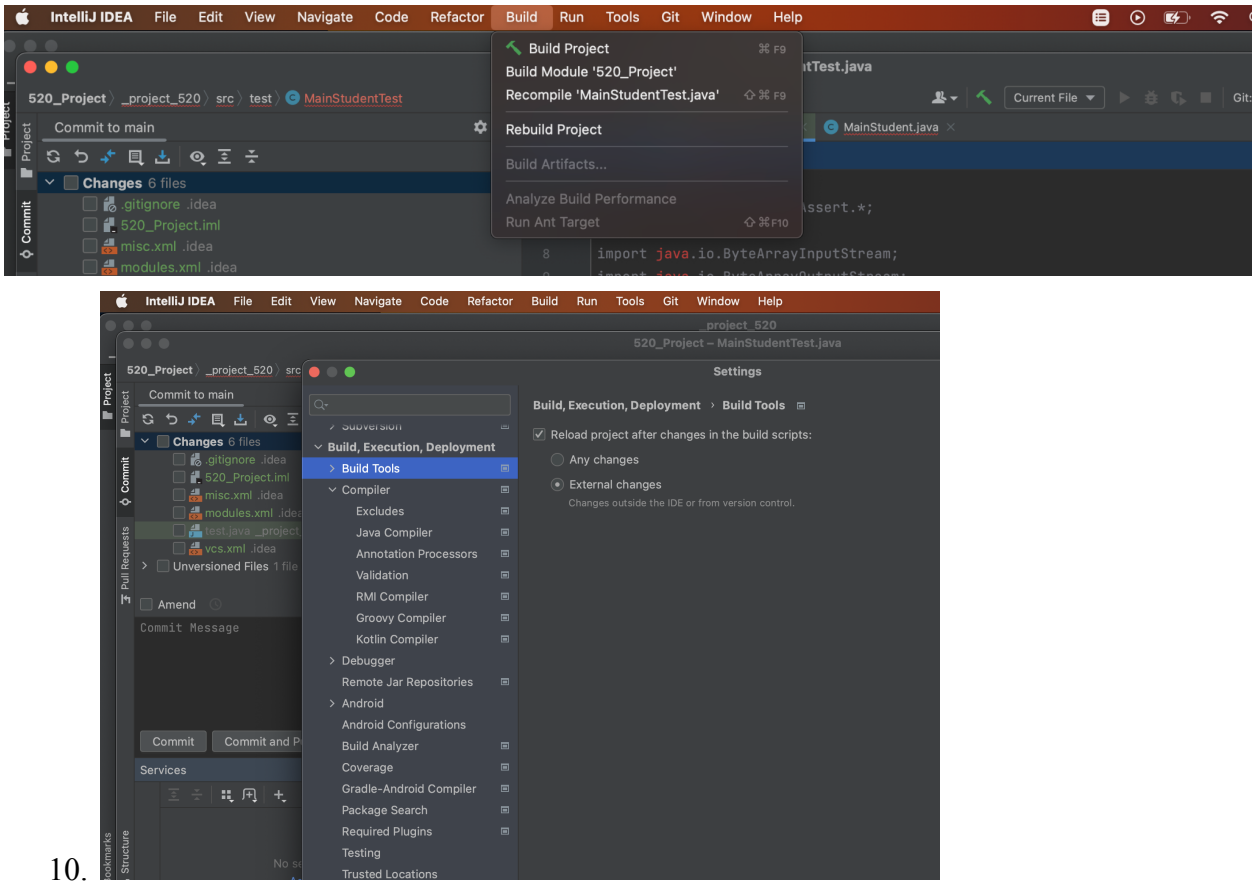
5. Build results and output are displayed within the IDE, allowing for easy debugging and error identification.
6. IntelliJ IDEA offers advanced build tools and features, including incremental compilation, dependency management, and build profiles.
7. Continuous integration and build servers can be seamlessly integrated with IntelliJ IDEA for automated builds.
8. The IDE provides build-related inspections and quick fixes to ensure code quality during the build process.
9. Build automation in IntelliJ IDEA enhances developer productivity and enables a smoother development workflow.
10. 

Fig: IntelliJ IDEA Build Automation

Visual Studio Code (VSCode):

Visual Studio Code's build integration empowers developers to automate code building and compilation directly within the IDE. It offers a seamless workflow for executing build tasks and examining build output, eliminating the necessity of switching to external build tools or command-line interfaces.Here's are a few features of build integration in Visual Studio Code:

1. Task Configuration: Visual Studio Code utilizes a tasks.json file where you can define build task configurations. This file specifies the commands and arguments required to perform the build.
2. Task Runner: The IDE features a built-in task runner that allows you to define and execute tasks such as building, compiling, testing, and running your code. The task runner can be accessed through the command palette or via keyboard shortcuts.
3. Build Tool Support: Visual Studio Code integrates with popular build tools such as Maven, Gradle, Gulp, Grunt, and more. It automatically detects the presence of build configuration files and provides features for managing dependencies, executing build tasks, and displaying build output.
4. Output Display: The IDE displays the build output in an integrated terminal, providing real-time feedback on the build progress, errors, warnings, and other relevant information. The output can be easily viewed and monitored within the Visual Studio Code interface.
5. Run Configuration: Visual Studio Code allows you to create run configurations for executing your code. These configurations define parameters, command-line arguments, environment variables, and other settings needed for running your application.
6. Error Highlighting: The IDE performs real-time analysis of your code and highlights errors, warnings, and other issues as you type. This helps catch potential build or compilation issues early in the development process.

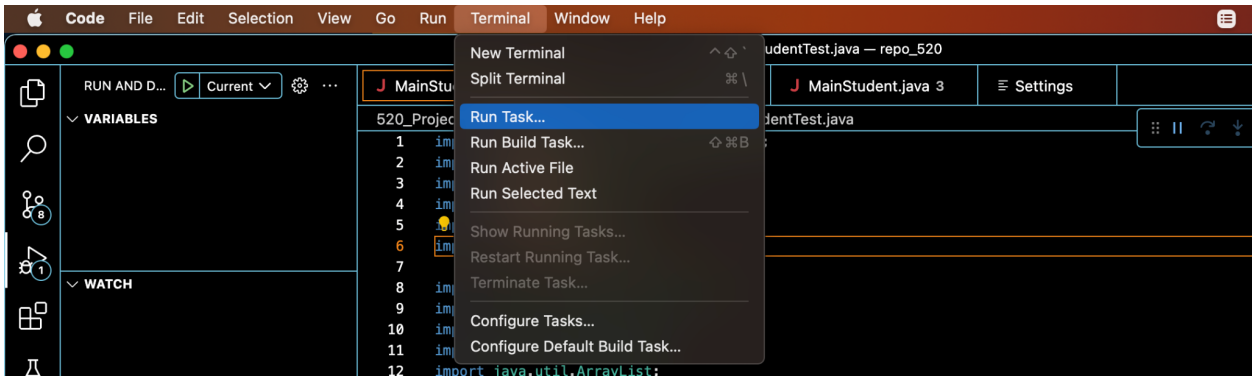


Fig: VS Code Build automation

- ## Extensibility

  If you are planning to transition from a code editor to a more feature-filled IDE then both of these options have plenty to offer. One of the important aspects to consider is extensibility. Extensions come by way of plug-ins or add-ons, and having relevant additions is a quick and efficient way to enhance your coding experience.

  IntelliJ IDEA:

- IntelliJ IDEA provides a highly extensible platform that allows developers to customize and extend the IDE's functionality.
- Developers can enhance IntelliJ IDEA by installing plugins from the IntelliJ Plugin Marketplace, which offers a wide range of plugins for different programming languages, frameworks, and tools.
- The IDE supports various extension points and APIs, allowing developers to create their own plugins and extensions.
- IntelliJ IDEA provides a rich set of APIs for interacting with the editor, accessing project structures, analyzing code, and integrating with external tools and services.
- Plugins can add new language support, including syntax highlighting, code completion, refactoring, code analysis, and more.
- Developers can extend the IDE with additional tools and features, such as version control integration, build automation, task runners, and database tools.
- IntelliJ IDEA supports creating custom inspections, intentions, and code generators to automate repetitive tasks and improve code quality.
- The IDE offers a flexible user interface that can be customized and rearranged according to individual preferences.
- Developers can create their own custom tool windows, panels, and views to display specialized information and tools.
- IntelliJ IDEA supports theming and provides options for customizing the editor's appearance, including syntax highlighting colors, font settings, and UI themes.
- The IDE includes a powerful plugin development environment with tools for debugging, testing, and packaging plugins.
- The IntelliJ Plugin Development Kit (PDK) provides extensive documentation and resources to help developers create high-quality plugins.
- The IntelliJ IDEA community is active and vibrant, with a large number of plugins and extensions developed and maintained by the community.
- IntelliJ IDEA's extensibility allows developers to tailor the IDE to their specific needs, improving productivity and enhancing the development experience.

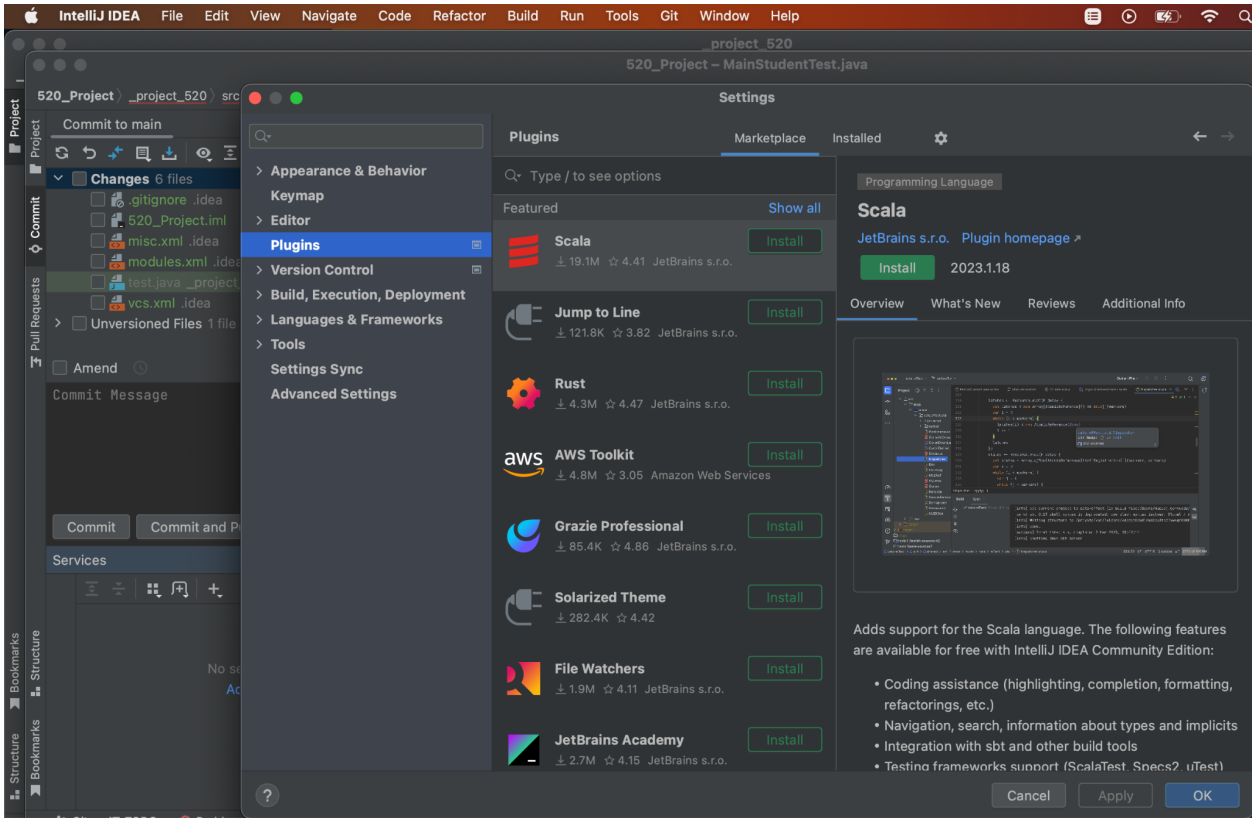


Fig: IntelliJ IDEA extensibility

Visual Studio Code (VSCode):

- Visual Studio Code (VS Code) offers extensive extensibility options, allowing developers to customize and enhance the functionality of the editor.
- VS Code extensions are available through the Visual Studio Code Marketplace, offering a wide range of features, tools, and language support.
- Developers can install extensions directly from within the editor, making it easy to extend its capabilities.
- VS Code provides a rich API that allows extension developers to interact with the editor, access and manipulate files, modify the user interface, and integrate with external tools and services.
- Extensions can add new language support, including syntax highlighting, code completion, linting, formatting, and more.
- Developers can enhance their debugging experience by installing debuggers for specific languages or frameworks.
- VS Code extensions can provide additional productivity tools, such as code snippets, project templates, and task automation.
- The editor supports themes and color schemes, and developers can install extensions to customize the editor's appearance and create their own themes.

- Extensions can integrate with popular source control systems, providing version control functionalities directly within the editor.
- Developers can extend the editor with additional features, such as code refactoring, code navigation, code analysis, and documentation generation.
- VS Code extensions can integrate with external services and APIs, enabling developers to interact with cloud platforms, databases, deployment tools, and more.
- The VS Code extension ecosystem is highly active, with a large community of developers contributing and maintaining a wide variety of extensions.
- Extensions can be developed using various programming languages, such as JavaScript, TypeScript, and Python, leveraging the power of Node.js and web technologies.
- VS Code provides tools and utilities for extension development, including a built-in extension debugger and a rich set of APIs and libraries.
- Developers can create their own extensions to tailor the editor to their specific needs or contribute to existing extensions to improve functionality for the entire community.
- The extensibility of VS Code makes it a versatile and adaptable editor that can be customized and extended to fit different programming languages, frameworks, and workflows.

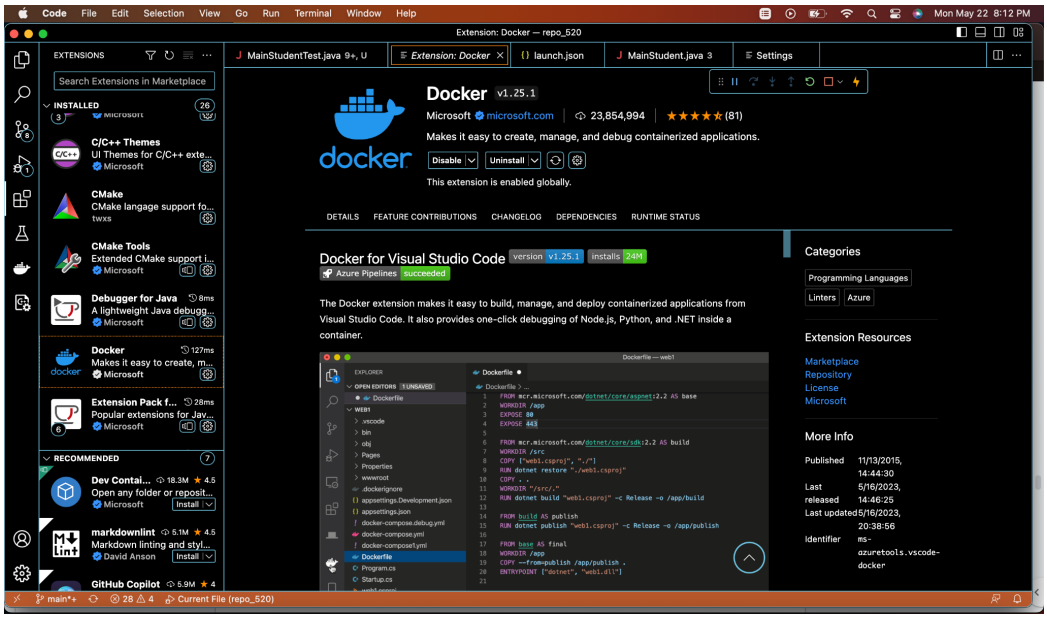


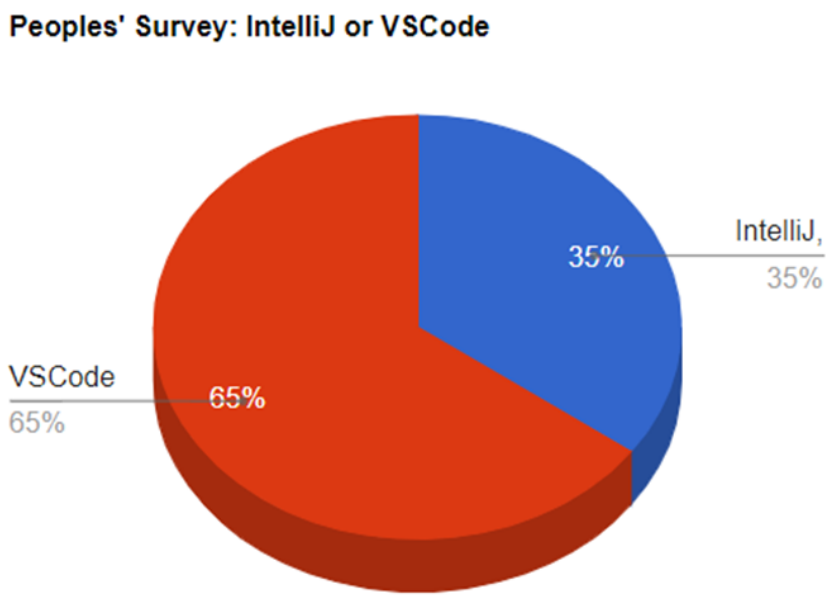
Fig: VS Code extensibility

Comparison Summary

	IntelliJ IDEA	**Visual Studio Code (VSCode)**
Language Support	Extensive support for various languages	Support for multiple languages through extensions
Code Completion	Advanced and context-aware suggestions	IntelliSense with customizable snippets
Refactoring	Rich set of refactoring tools and features	Limited refactoring capabilities
Debugging	Robust debugging capabilities	Built-in debugging support
Extensions	Limited extension ecosystem	Vast extension marketplace
Learning Curve	Steeper learning curve due to feature depth	Relatively easier to learn and get started
IDE Features	Comprehensive tools and productivity features	Lightweight with basic features
Maintainability	Developed by JetBrains, regular updates	Supported by Microsoft, frequent updates
Stability	Generally stable, occasional issues	Generally stable, influenced by extensions
Support	JetBrains support plans available	Community support and forums

Documentation	Extensive official documentation	Comprehensive official documentation

Survey metrics in a Pie Chart:



Conclusion:

As the name implies, VS Code is focussed on code. The tool is very slim in terms of footprint of features in the UI. Its usability concept is centered around using the keyboard, rather than the mouse. Many features are available via CLI or the command palette only. In fact, some claim VS Code to be a hybrid between an IDE and a simple code editor. This reflects the preference of many developers nowadays, especially for web development.

Related Works

- Tools reference:
  IntelliJ IDEA: https://www.jetbrains.com/help/idea/getting-started.html
  Visual Studio Code: https://code.visualstudio.com/docs

- Papers referred:
  - A Comparison of Project Management Software Tools (PMST)
  By Halil Cicibas, Kadir alpaslan Demir (Published in 2010)

  - A comparative study of software tools for user story management
  By Sonja Dimitrijević, Jelena Jovanović, Vladan Devedžić

  - Comparing software metrics tools
  By Rüdiger Lincke, Jonas Lundberg, Welf Löwe

- Websites referred for Student Attendance Management Portal code:
  - sourcecodehero.com/student-management-system-project/
  - GeeksForGeeks