Mini Project Guide

1. Install Node.js and Git on your system, if you don’t have them already. You will need them to run the VS Code extension generator and to publish your extension later.
2. Install Yeoman and VS Code Extension Generator with the command npm install -g yo generator-code. These are tools that will help you scaffold a basic extension project with the necessary files and configurations.
3. Run the generator with the command yo code and answer the prompts. Choose New Extension (TypeScript) as the type of extension, and give it a name, an identifier, and a description. You can also choose to initialize a Git repository, bundle the source code with webpack, and select a package manager (npm or yarn). Finally, you can choose to open the new folder with VS Code or not.
4. Inside VS Code, open the src/extension.ts file and press F5 to run the extension in a new Extension Development Host window. This will compile and run your extension in a separate instance of VS Code, where you can test it and debug it.
5. To use an API of an open source AI that corrects C language code, you will need to find one that suits your needs and preferences. For example, you can use [SourceAI](https://sourceai.dev/), which is an AI-powered tool that can generate code in any programming language from any human language description. It can also simplify, find errors and fix them, and debug your code. You will need to sign up for an account and get an API key to use their service.
6. To call the API from your extension, you will need to use the https module in Node.js, which allows you to make HTTP requests. You can import it at the top of your extension.ts file with the statement import \* as https from 'https';.
7. To interact with the user, you will need to use the vscode module, which provides various APIs for VS Code extensions. You can import it at the top of your extension.ts file with the statement import \* as vscode from 'vscode';.
8. To register a command for your extension, you will need to use the vscode.commands.registerCommand function, which takes a command identifier and a callback function as arguments. You can do this inside the activate function of your extension.ts file, which is called when your extension is activated. For example, you can register a command called extension.correctCCode with the following code:

export function activate(context: vscode.ExtensionContext) {  
 let disposable = vscode.commands.registerCommand(  
 'extension.correctCCode',  
 async () => {  
 // Your code here  
 }  
 );  
  
 context.subscriptions.push(disposable);  
 }

1. Inside the callback function of your command, you can write the logic for calling the API and displaying the results to the user. For example, you can do the following steps:

* Get the active text editor and the selected text from it using the vscode.window.activeTextEditor and vscode.window.activeTextEditor.document.getText(selection) properties.
* Check if the selected text is valid C code using a regular expression or a parser library.
* If not, show an error message to the user using the vscode.window.showErrorMessage function.
* If yes, make a POST request to the SourceAI API endpoint with your API key and the selected text as parameters using the https.request function.
* Handle the response from the API using the response.on function and parse it as JSON using the JSON.parse function.
* If there is an error from the API, show an error message to the user using the vscode.window.showErrorMessage function.
* If there is a result from the API, show it to the user using the vscode.window.showInformationMessage function or replace the selected text with it using the vscode.window.activeTextEditor.edit function.

1. Test your extension by running it in the Extension Development Host window and selecting some C code in a file. Invoke your command from the Command Palette (Ctrl+Shift+P) or assign a keyboard shortcut to it in your package.json file under the contributes.keybindings section.
2. Publish your extension to the VS Code Marketplace using the vsce publish command after creating an account and getting a personal access token from [here]. You can also package your extension as a .vsix file using the vsce package command and distribute it manually.

REFERENCES-

1.[VS Code Extension API documentation](https://code.visualstudio.com/api/get-started/your-first-extension) and [SourceAI documentation](https://sourceai.dev/).

2.[Kodezi](https://kodezi.com/), [OpenAI Codex](https://openai.com/blog/openai-codex/), or [Code Fixer.](https://codepal.ai/code-fixer)