**Assignment:** Snap-in Development Challenge Submission

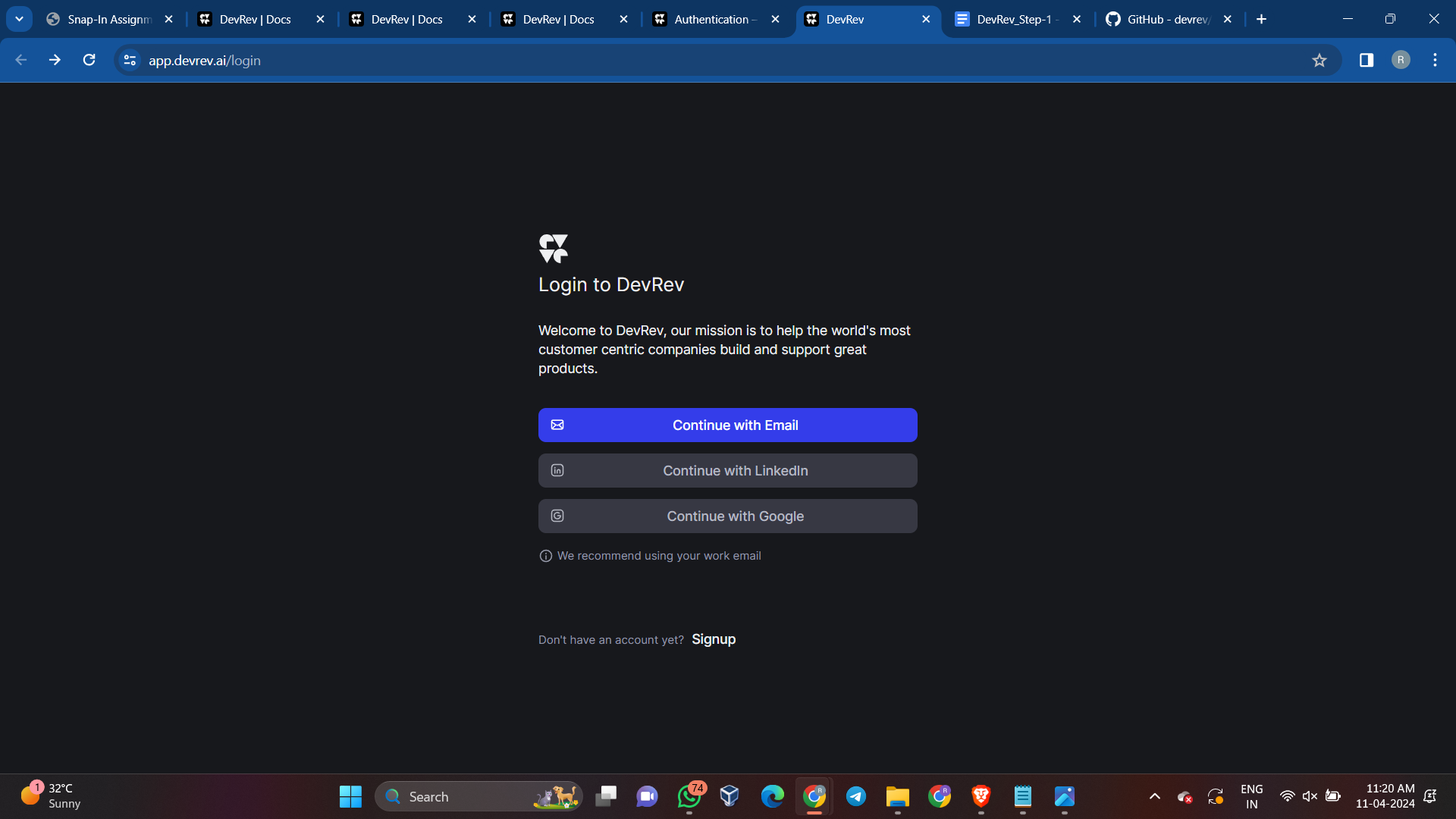
## **Step 1** : Utilizing the DevRev AP

First task is to utilize the DevRev API to create a work item. Python was used as the programming language, which is compatible with the DevRev API. The Code snippet demonstrates the creation of a work item using the DevRev API.

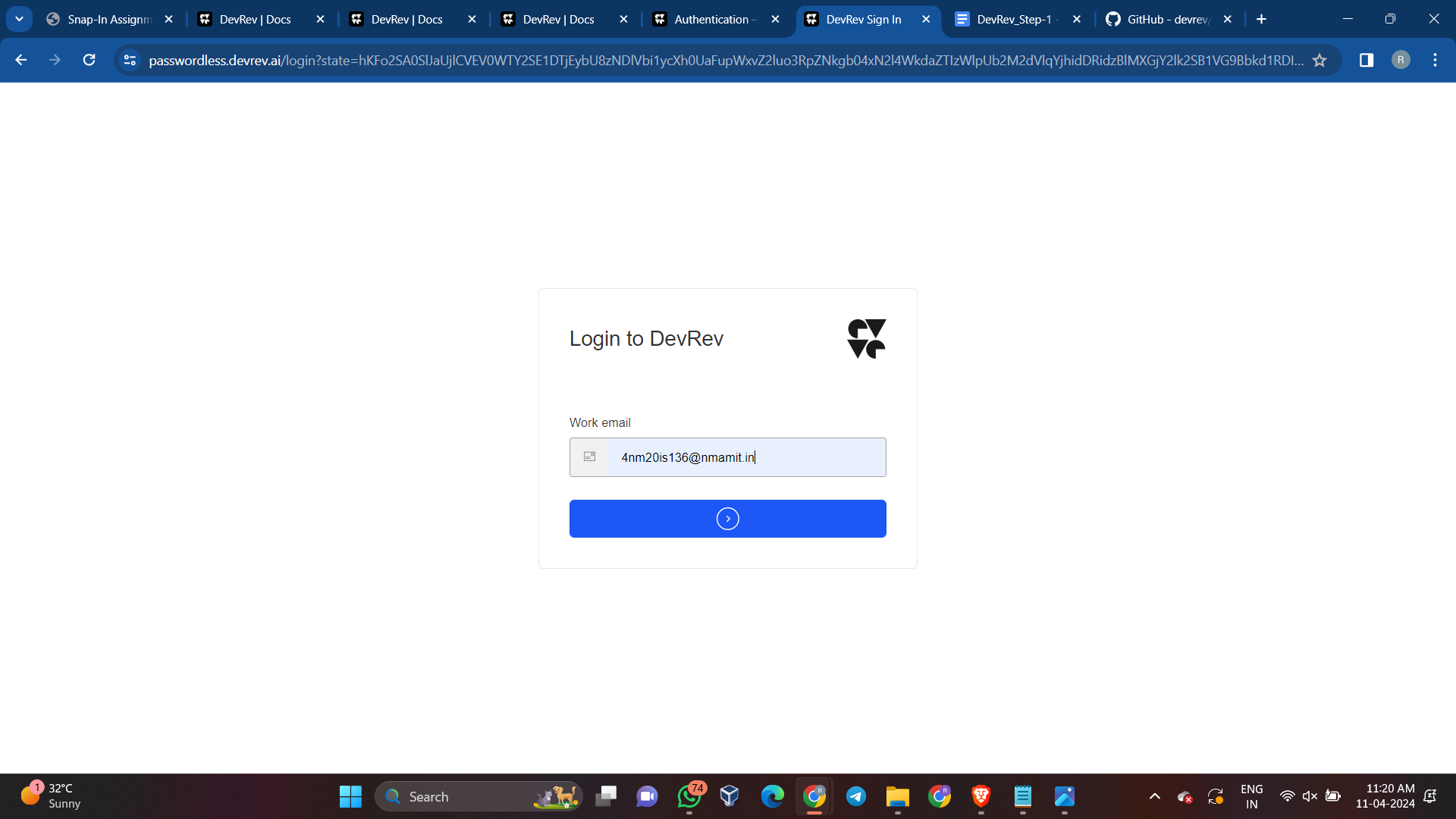
**Initial Steps to be executed:**

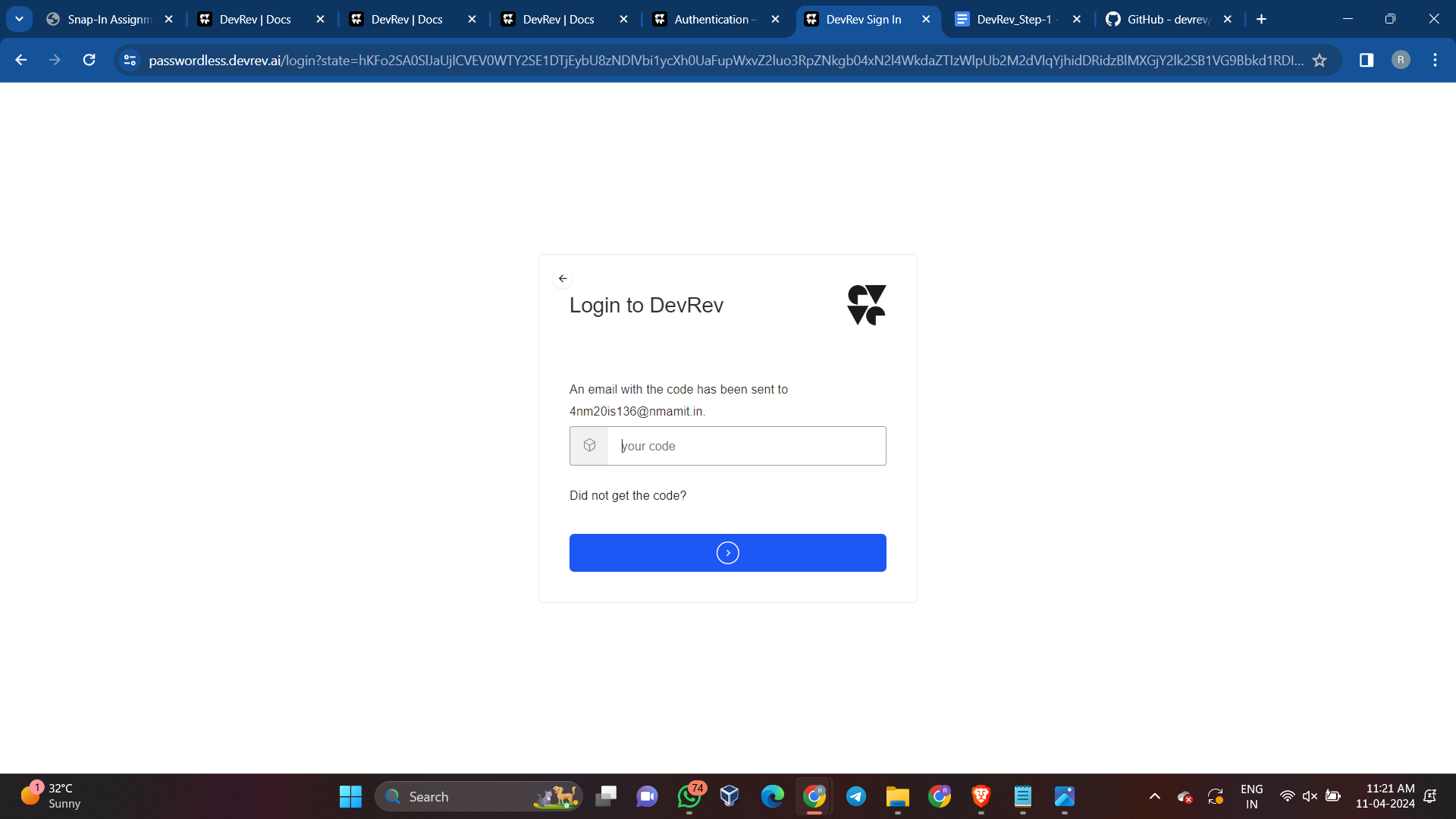
1. Creation of DevRev account:

* Click on the sign in option
* Continue with Gmail

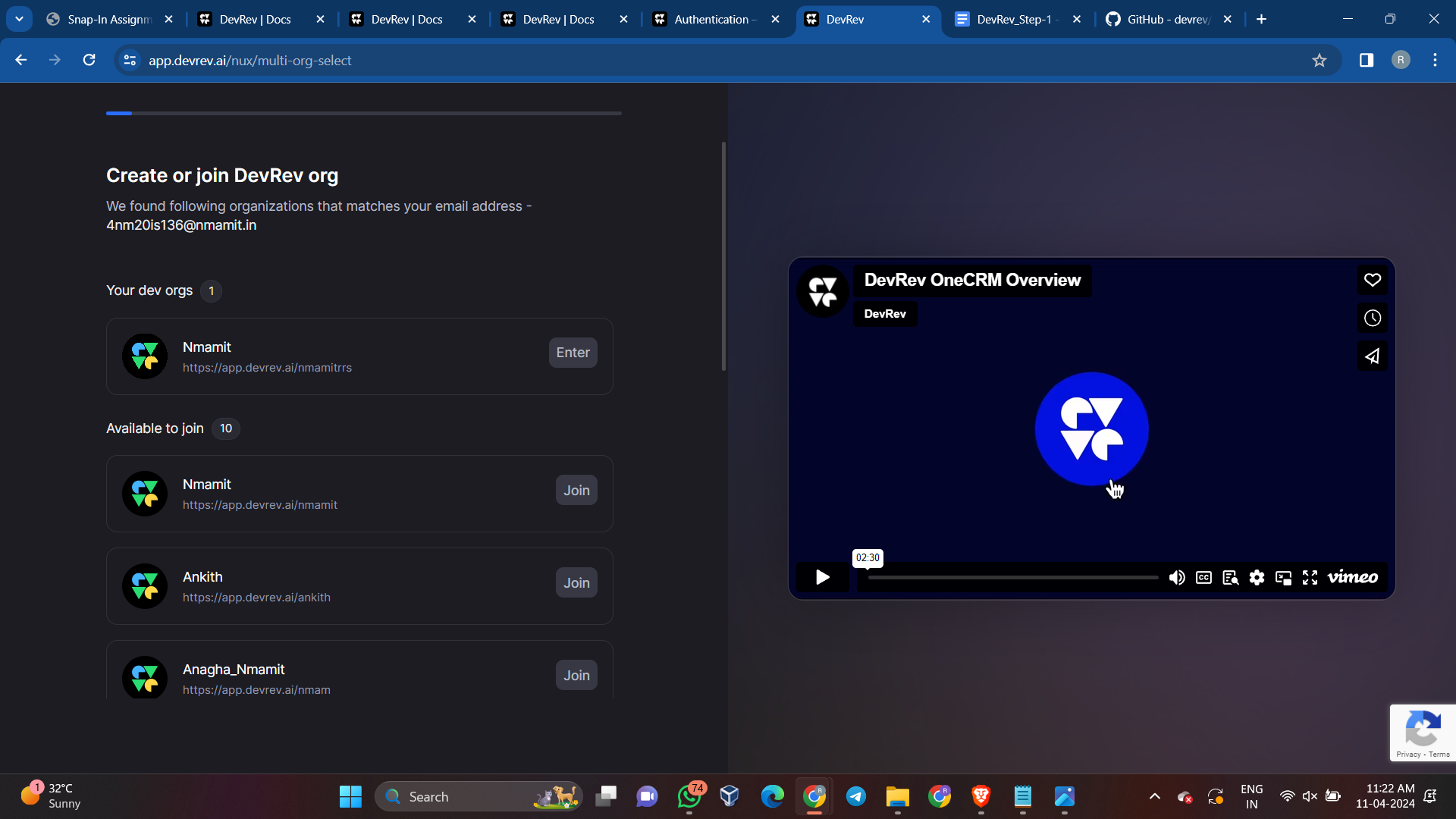


* Enter Work email: here college email id and enter the verification code sent on your email id entered.



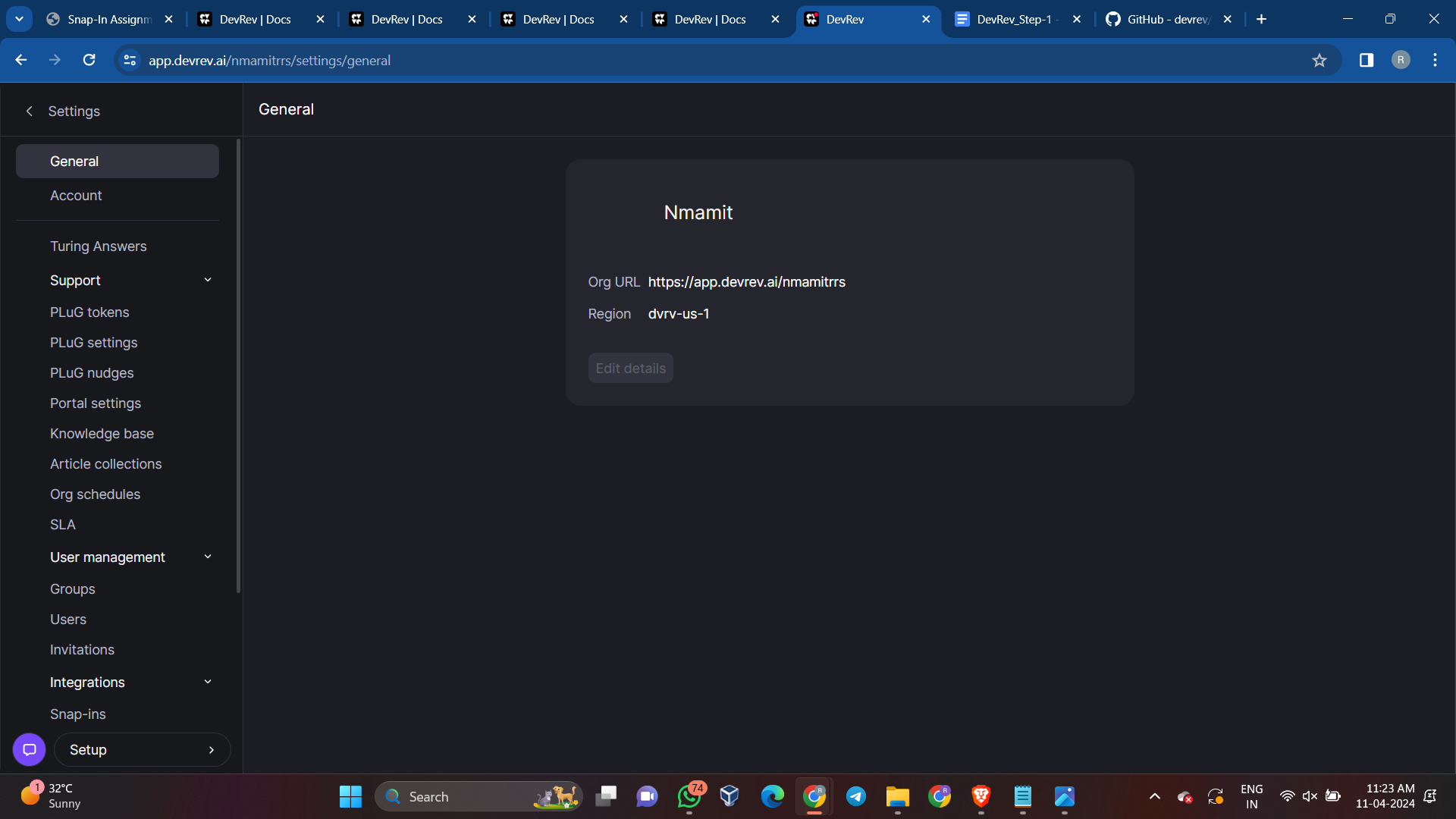


* Create a workspace or if already created, enter your workspace

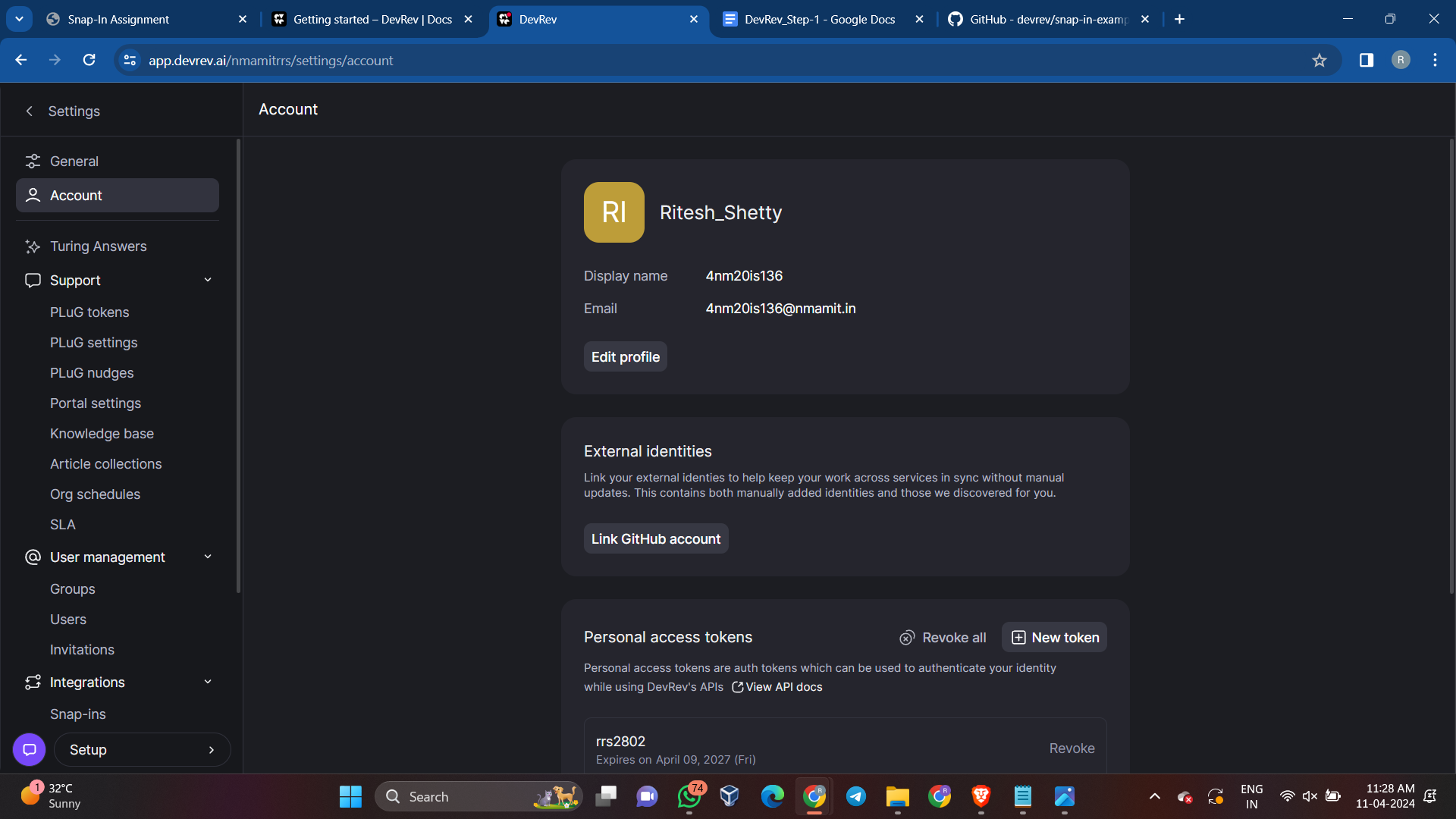


1. Generating your PAT(Personal access token):

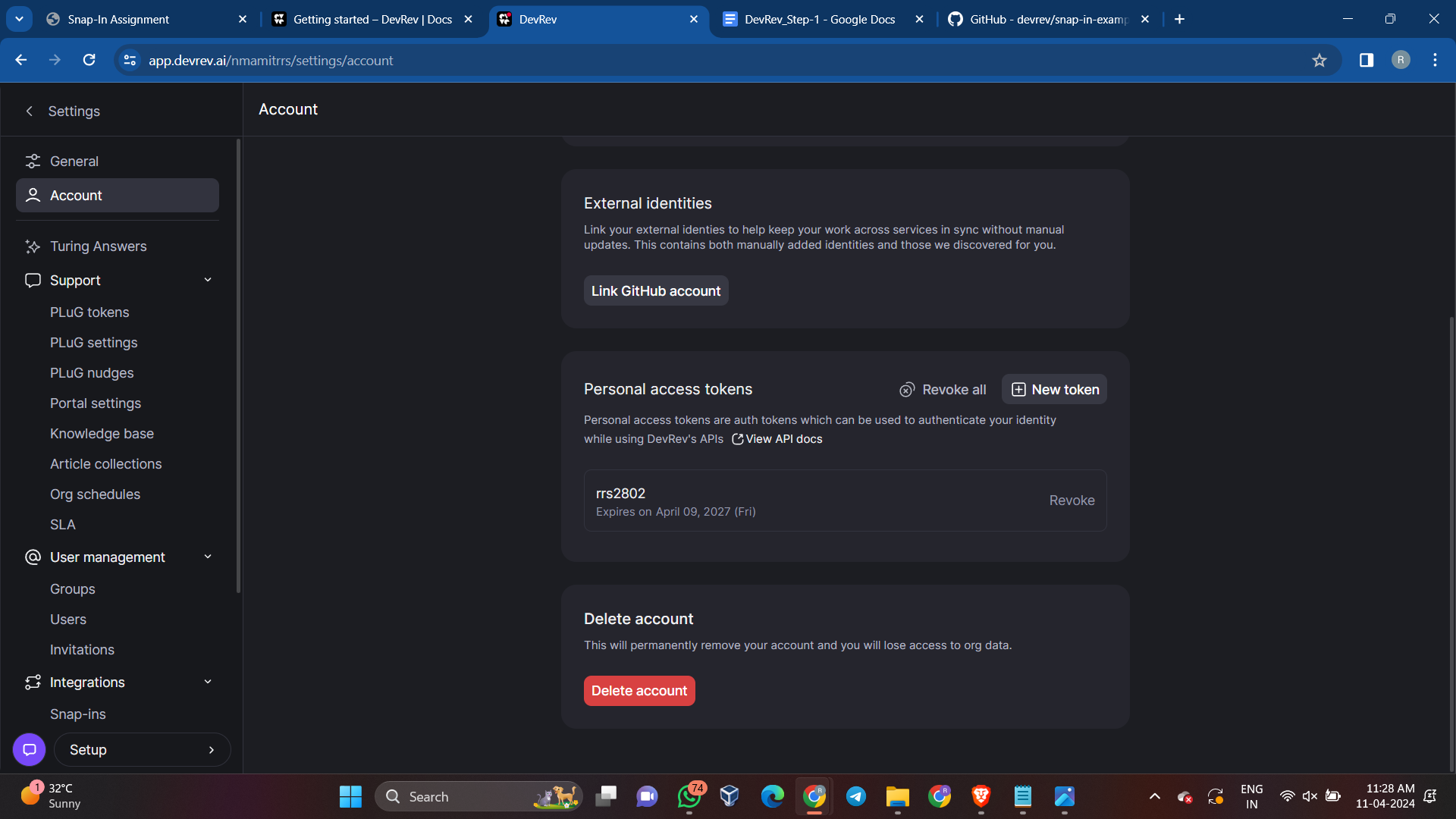
* Go to the relevant dev org in the DevRev app and navigate to Settings > Account > Personal Access Token to create a PAT.

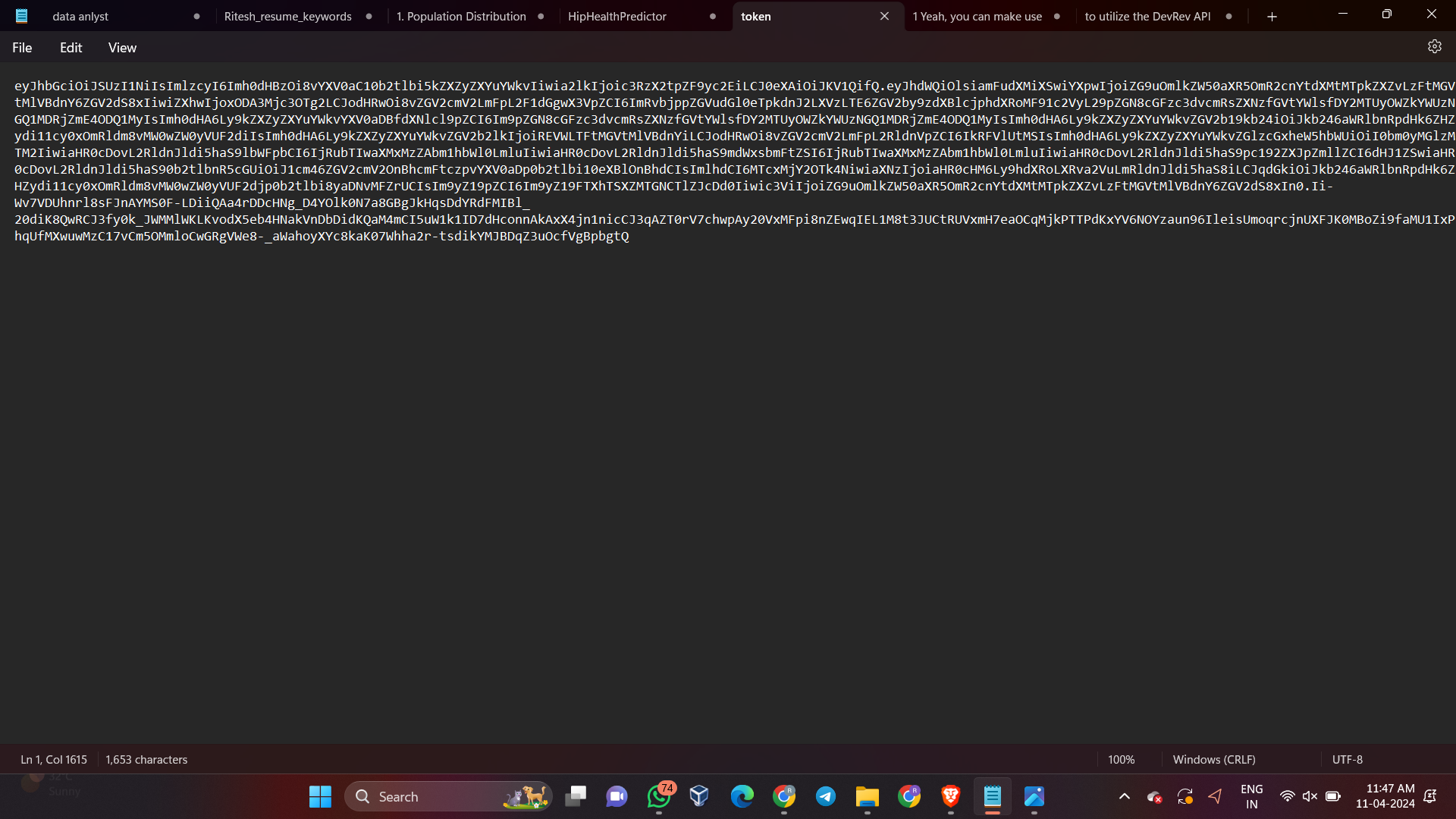


* Use descriptive names for PATs to differentiate between them, as the token value can't be retrieved later.



* After creating the PAT, securely store it, as it can't be retrieved once you leave the page.





**Brief explanation of what is PAT:**

A personal access token (PAT) is used for authenticating to DevRev APIs and uniquely identifying a dev user within a dev org. It allows external third-party applications to access DevRev APIs on behalf of the corresponding dev user, with privileges matching those of the token owner. The validity duration of a PAT can be set, but it cannot be renewed; instead, a new PAT must be created and code updated to use it. For example, a VS Code plugin may rely on a user's PAT to authenticate and access DevRev APIs for specific tasks.

**Explanation on what are Work Items:**

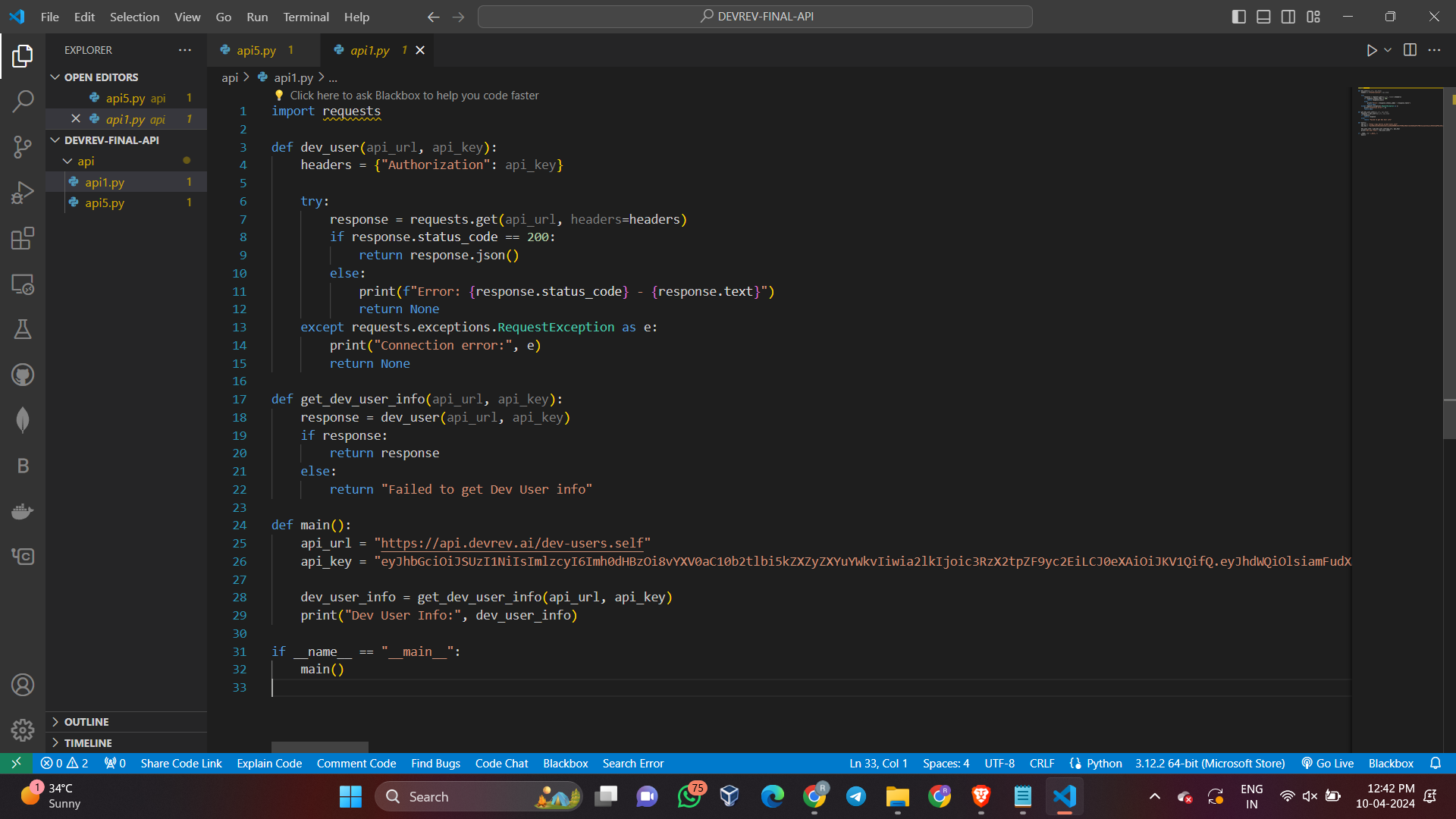
A work item within DevRev is any artifact that necessitates action from a human or machine and is associated with owners and effort. Work items can be linked together, forming parent/child relationships, and can lead to other work of the same or different type. Tickets, issues, enhancements, and tasks are specific types of work items, each with distinct creation contexts and purposes.

* Tickets are created by customers or consumers and are part of the Support app.
* Issues are created by builders or maintainers and are part of the Build app.
* Enhancements serve as parent items for multiple issues, leading to desired product changes.
* Tasks are used to break down larger work into smaller pieces.
* DevRev's "Similar work" model helps prevent duplicate work during creation by identifying potential duplicates and allowing for linkage to related work items.

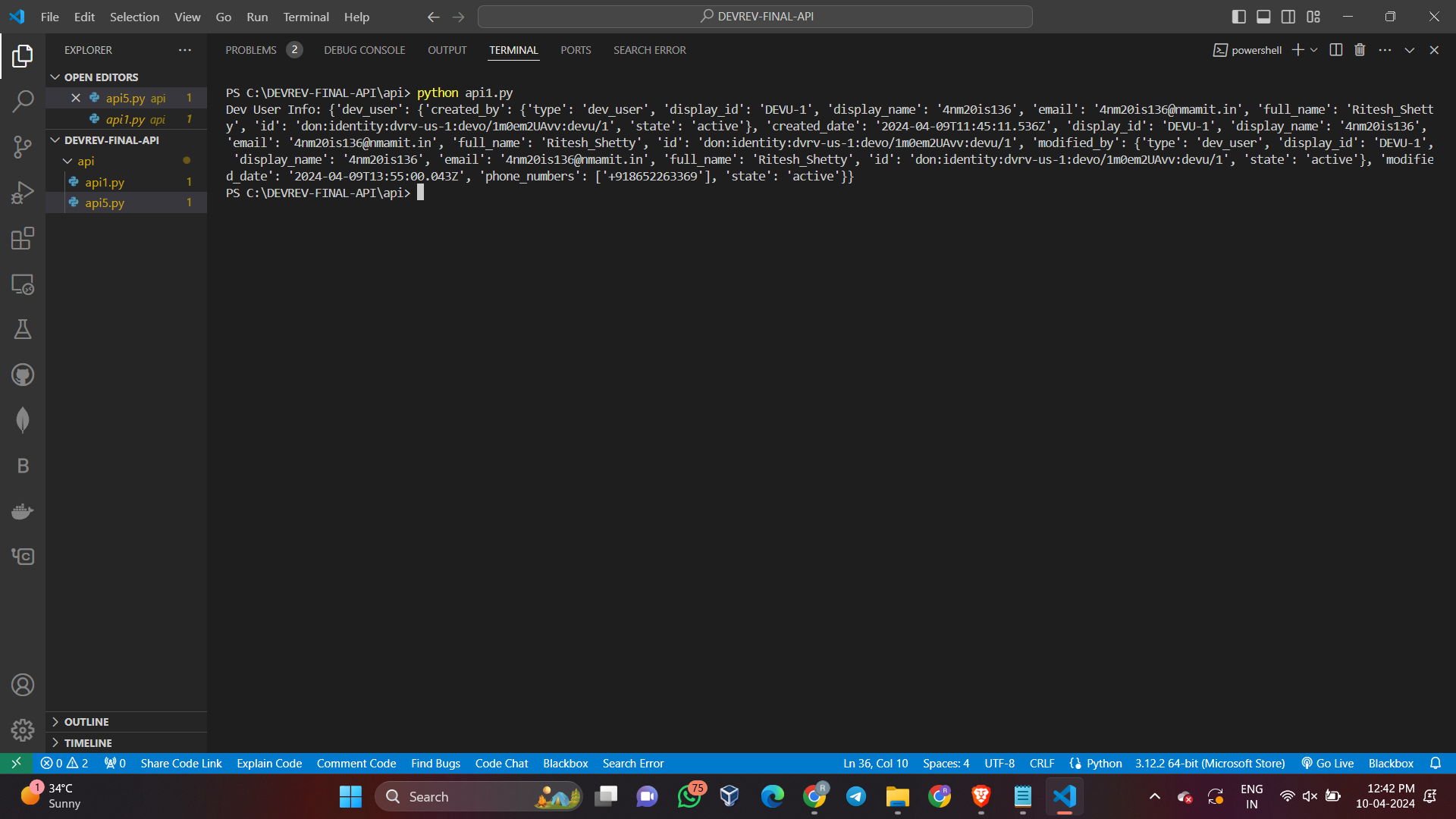
**Code snippet and Explanation to utilize the DevRev API to create a work item:**

**Explanation for api1.py:**

* The provided code makes use of the "requests" library to perform an HTTP GET request to a specified API endpoint which is to retrieve the user details using the unique API key.
* The "dev\_user" function is defined to make an HTTP GET request to the specified API endpoint using the provided API key for authorization. It handles potential errors and returns the response in JSON format if the status code is 200.
* The "get\_dev\_user\_info" function calls the "dev\_user" function with the API URL and API key, and returns the response if it exists, otherwise it returns a failure message.
* The "main" function sets the API URL and API key, then calls "get\_dev\_user\_info" to retrieve and print the Dev User information.
* The script uses the API URL "https://api.devrev.ai/dev-users.self" and a lengthy API key for authentication.
* The script then executes the "main" function if it's the main module, triggering the retrieval and display of the User information.

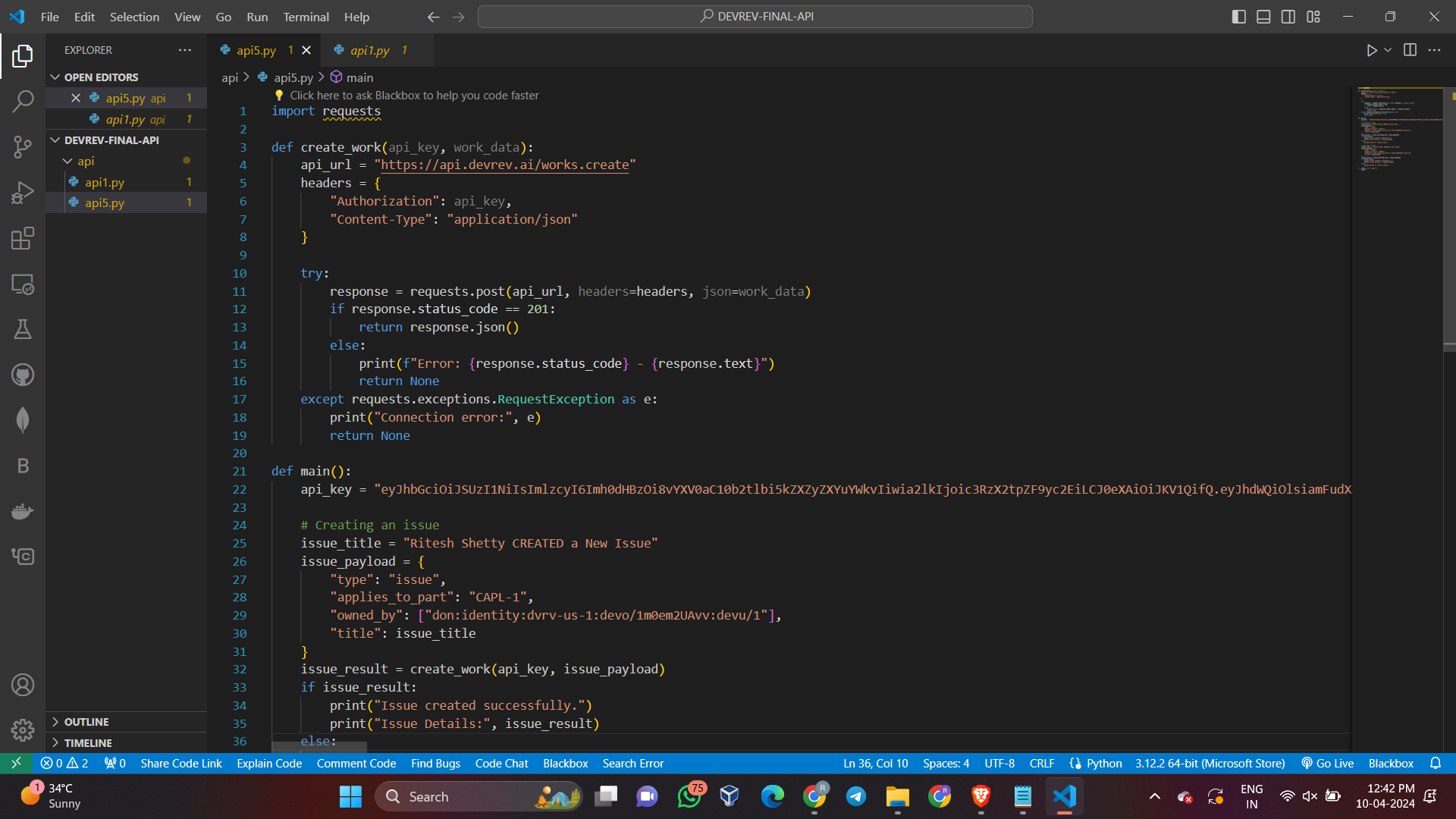


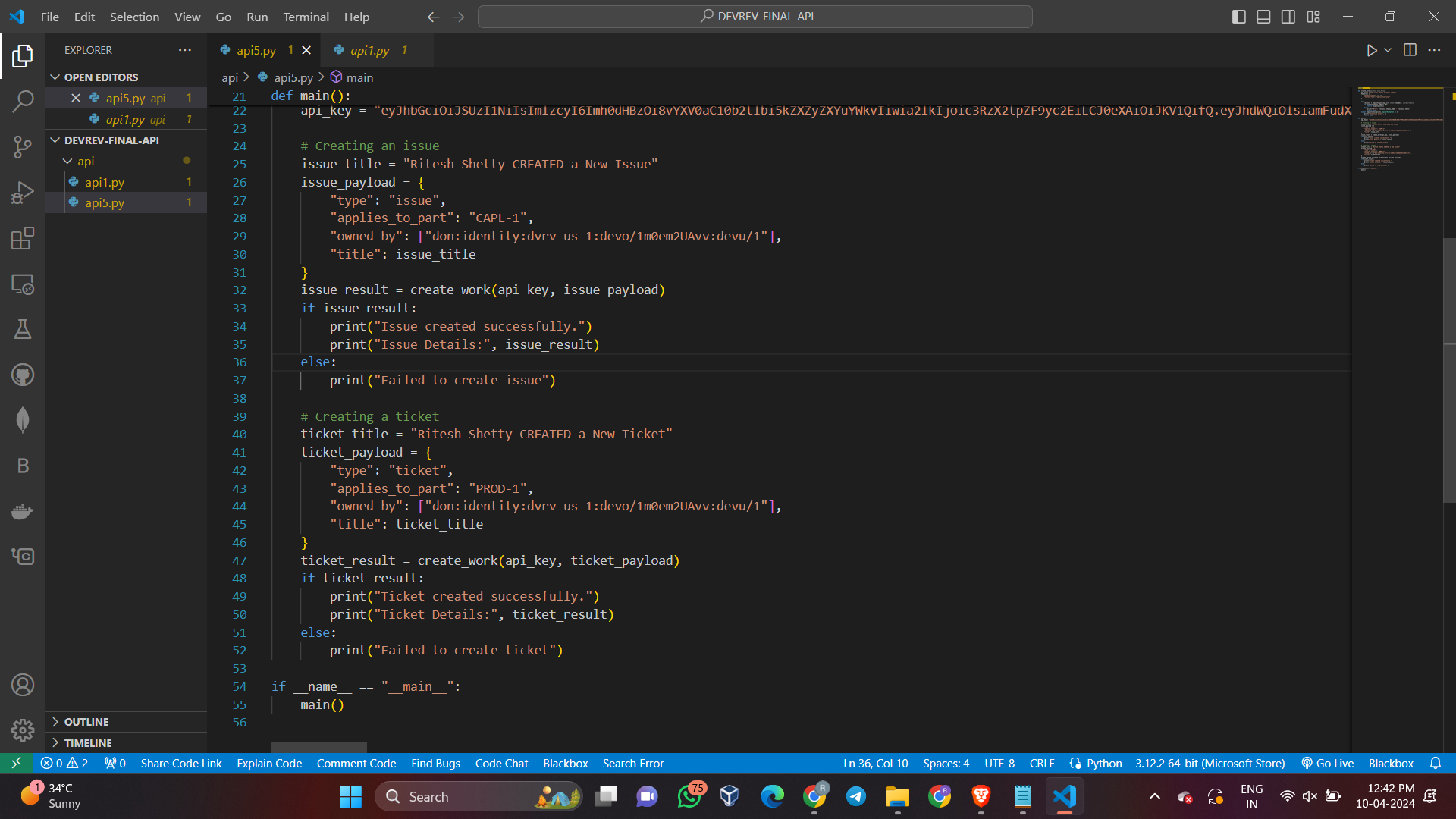
**Output:**

****

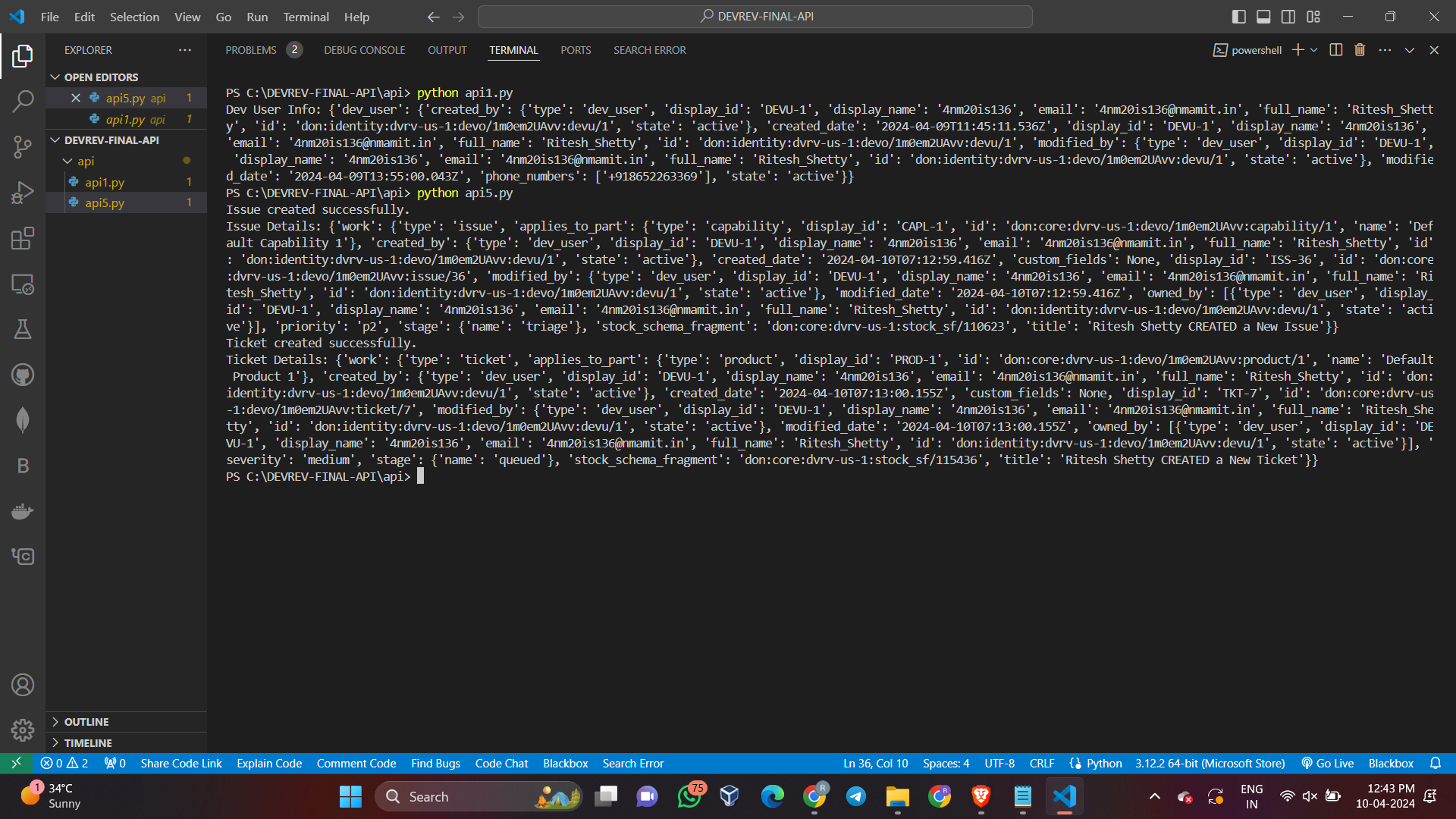
**Explanation for api5.py:**

* Importing the requests module: The code begins by importing the requests module, which allows the Python program to send HTTP requests.
* Defining the create\_work function: This function is responsible for creating a work item using the DevRev API. It takes an API key and work data as input, constructs the API URL, sets the required headers, and then makes a POST request to create the work item.
* Main function: The main function is defined, which serves as the entry point of the program. It initializes the API key and then proceeds to create an issue and a ticket using the create\_work function.
* API Key: The API key is a long string used for authentication and authorization to access the DevRev API which is unique to every user.
* Creating an issue: A sample issue is created with a specific title, type, applies\_to\_part, and owner details. The issue payload is then passed to the create\_work function to create the issue via the DevRev API.
* Handling issue creation result: The result of the issue creation attempt is checked, and a success or failure message is printed based on the outcome.
* Creating a ticket: Similar to creating an issue, a sample ticket is created with specific details, and the ticket payload is then passed to the create\_work function to create the ticket via the DevRev API.
* Handling ticket creation result: The result of the ticket creation attempt is checked, and a success or failure message is printed based on the outcome.
* Executing the main function: The program then checks if the script is being run as the main program, and if so, it executes the main function.
* Error handling: The code includes error handling using try-except blocks to catch any potential connection errors when making requests to the DevRev API.

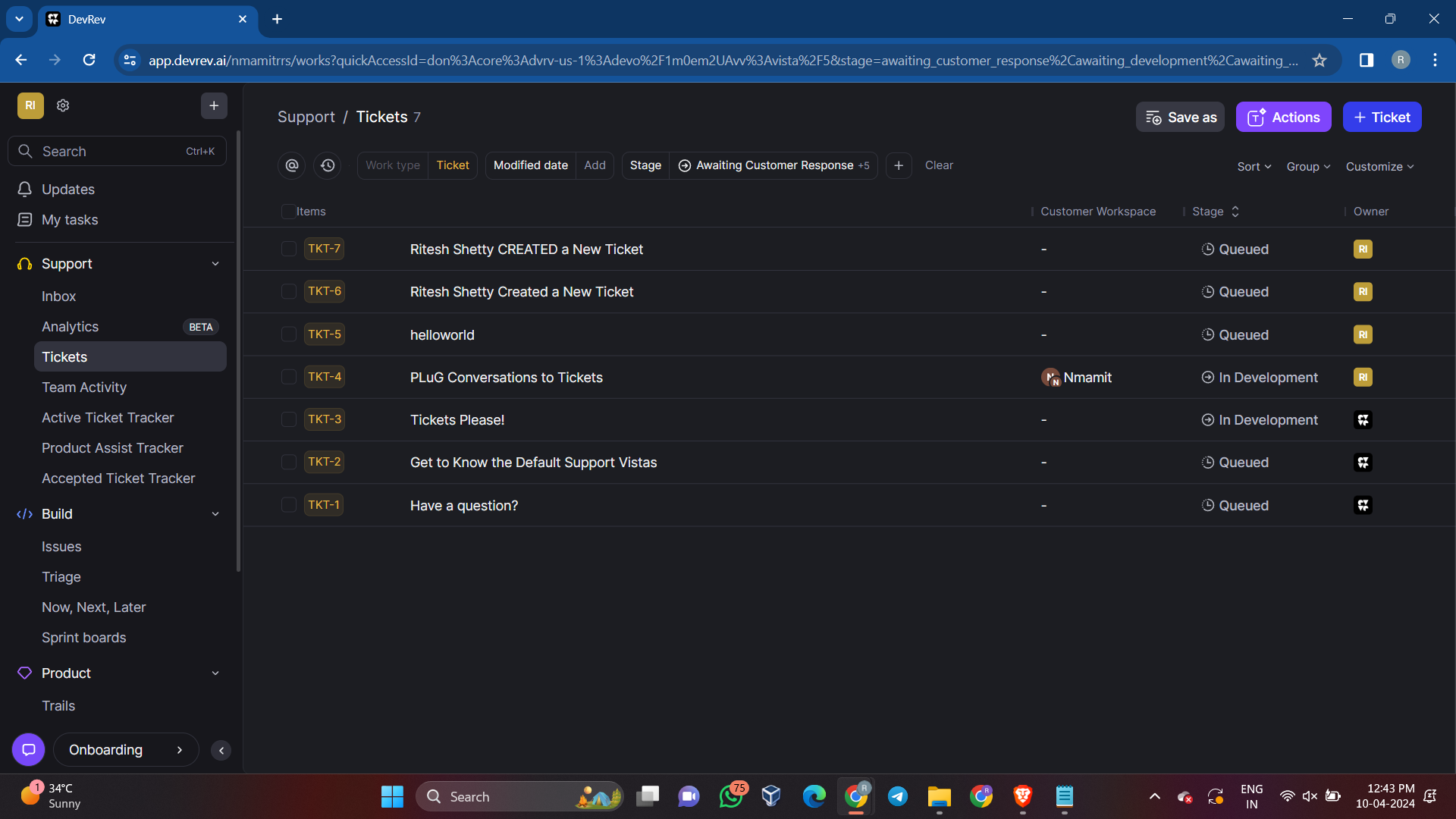


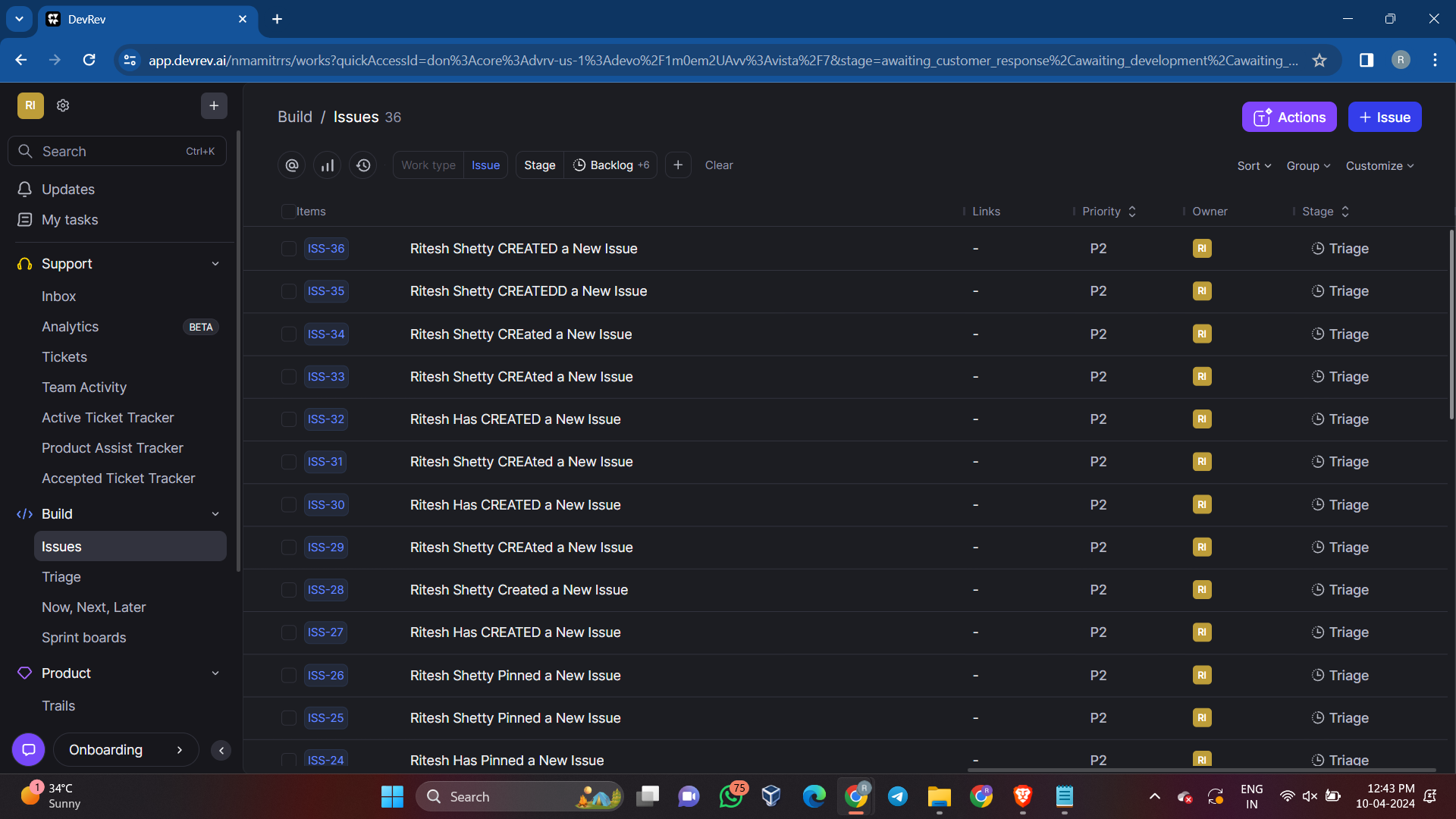


**Output:**

****

**Overall Output On the creation of Issues and Tickets on DevRev platform:**

****

****

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*