**SOFTWARE ENGINEERING LAB**

**TASK -1**

-K. ANIRUDH

- HU22CSEN0100610

What is GitHub?

GitHub is a cloud platform for version control and collaboration over the development of software. On top of Git, GitHub allows individuals and teams to manage, track, and collaborate on code projects more efficiently. GitHub builds up on Git with its extra features such as repository hosting, project management, and automation tools.

Key features of GitHub

**1. Version Control:-**

GitHub uses Git, a robust version control system, to track changes made to files over time. This feature allows developers to:

* Keep an exhaustive history of changes in their projects.
* Work on projects collaboratively without the fear of overwriting each other's work.
* Return to earlier versions of code when needed, hence stability in the development process.

**2. Repositories :-**

A repository is the central workspace of a project where all files, whether it is code or documentation, are kept. The repositories in GitHub allow for: - Secure cloud-based storage that makes projects accessible from anywhere.

* Both public and private repositories (with restrictions to collaborators).
* Projects are organized with features like README files for overviews and `.gitignore` files to manage unnecessary files.

**3. Branches:-**

Branches enable developers to work on independent versions of a project, allowing:

* Creation of new features or bug fixes without affecting the main codebase.
* Experimentation in isolated environments, with the ability to merge successful changes back into the main branch.
* Simultaneous collaboration, as multiple developers can work on different branches and later combine their work.

**4. Pull Requests:-**

* Pull requests are among GitHub's most powerful collaboration tools. They enable:
* Developers to suggest changes in a repository by comparing a branch with the main codebase.
* Teams to review and discuss the changes before integrating them into the main project.
* Automated checks to test changes that have been proposed for code and merge it, which would have been error-free before that.
* 5. Issues and Project Management
* Tools of GitHub help to manage tasks, track bugs and also prioritize work. Through issues, teams can;
* Bug reports, feature requests and even other tasks about the project could be documented.
* Set up tasks to specific people and deadlines by using milestones.
* Use project boards that resemble Kanban boards for visualizing and tracking work more effectively.

**6. Actions and Automation:-**

* Using GitHub Actions, developers can automate repetitive tasks and save developers' time and reduce error. Some of the major benefits are:
* Running automated tests and builds whenever new code is pushed to the repository.
* Deploying applications to staging or production environments automatically.
* Creating custom workflows tailored to a project's needs, which will enhance overall efficiency.

**Comparison between Git and GitHub**

| **Feature** | **Git** | **GitHub** |
| --- | --- | --- |
| **Type** | Open-source version control system (VCS). | Cloud-based platform for hosting Git repositories. |
| **Purpose** | Manages project history and tracks changes locally. | Facilitates remote storage, sharing, and collaboration. |
| **Installation** | Installed and run on the local machine. | Accessed via a web browser or Git client. |
| **Main Functionality** | Provides version control for local development. | Enhances collaboration with tools like pull requests and issues. |
| **Collaboration** | Requires manual setup for syncing changes. | Offers built-in collaboration features such as pull requests and issue tracking. |
| **Automation** | Does not include automation tools. | Includes automation via GitHub Actions for CI/CD tasks. |
| **Accessibility** | Can operate offline on a local system. | Requires internet for accessing repositories. |
| **Repository Location** | Stores files and history locally on your machine. | Hosts repositories remotely in the cloud. |
| **Interface** | Operated via command-line or Git GUI tools. | Provides a user-friendly web interface and project management tools. |

**Creating GitHub Account**

Making a GitHub account is direct. Fair take after these steps:

**Step 1:**

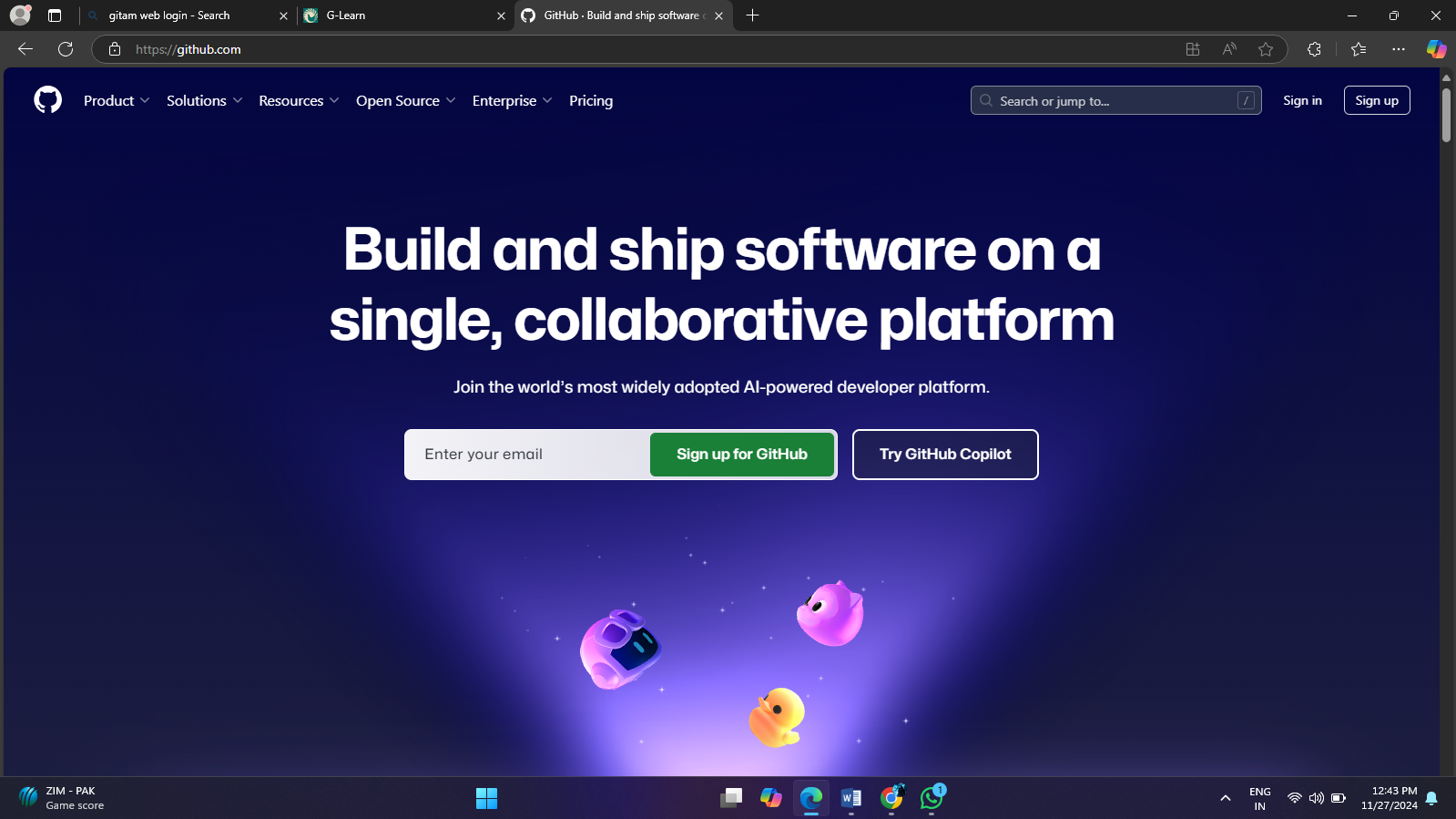
* Open the GitHub website
* Go to the GitHub homepage in a web browser.
* Click the Sign Up button in the top-right corner of the page.

**Step 2: Enter Your Information**

* Username: Select a one of a kind username that will be appeared on your GitHub profile, e.g., `MyGitHubUsername`.
* E-mail Address: Give a substantial e-mail address for account notices and recuperation.
* Secret word: Make a solid secret word to secure your account. Utilize a blend of capitalized, lowercase, numbers, and uncommon characters.

**Step 3: Total the Verification**

* Unravel the CAPTCHA or astound that GitHub presents to affirm you're not a bot.
* Press \*\*"Make Account"\*\* to proceed.

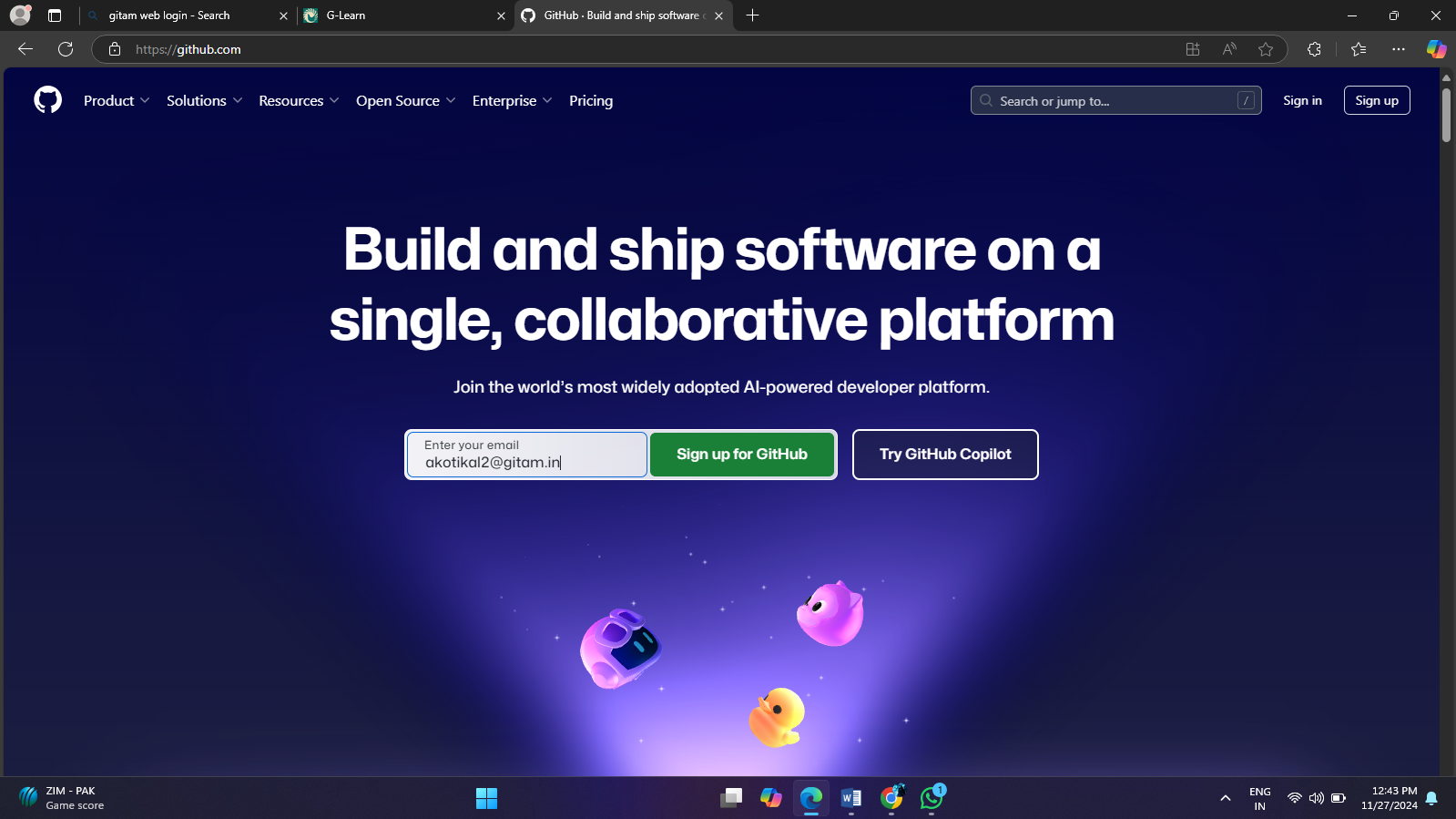


**Step 4: Select Your Plan**

* Select a \*\*Free Arrange\*\* for person utilize or investigate paid choices for group ventures.
* Tap "Proceed" after selecting your plan.

**Step 5: Affirm Your Email**

* Check your e-mail inbox for a affirmation e-mail from GitHub.
* 2. Open the mail and tap on the confirmation interface given. This actuates your account.



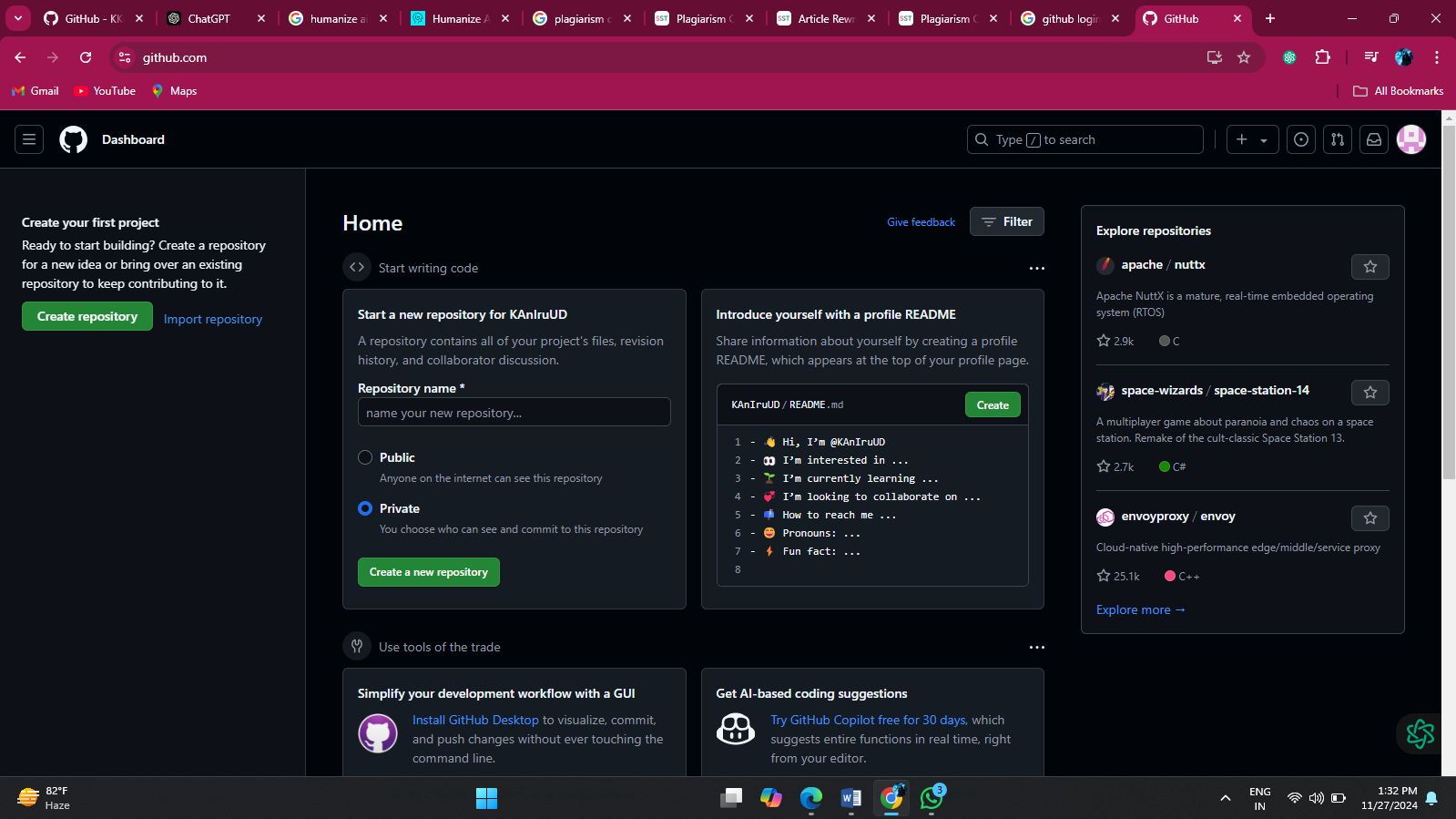
**Step 6: Customize Your Experience**

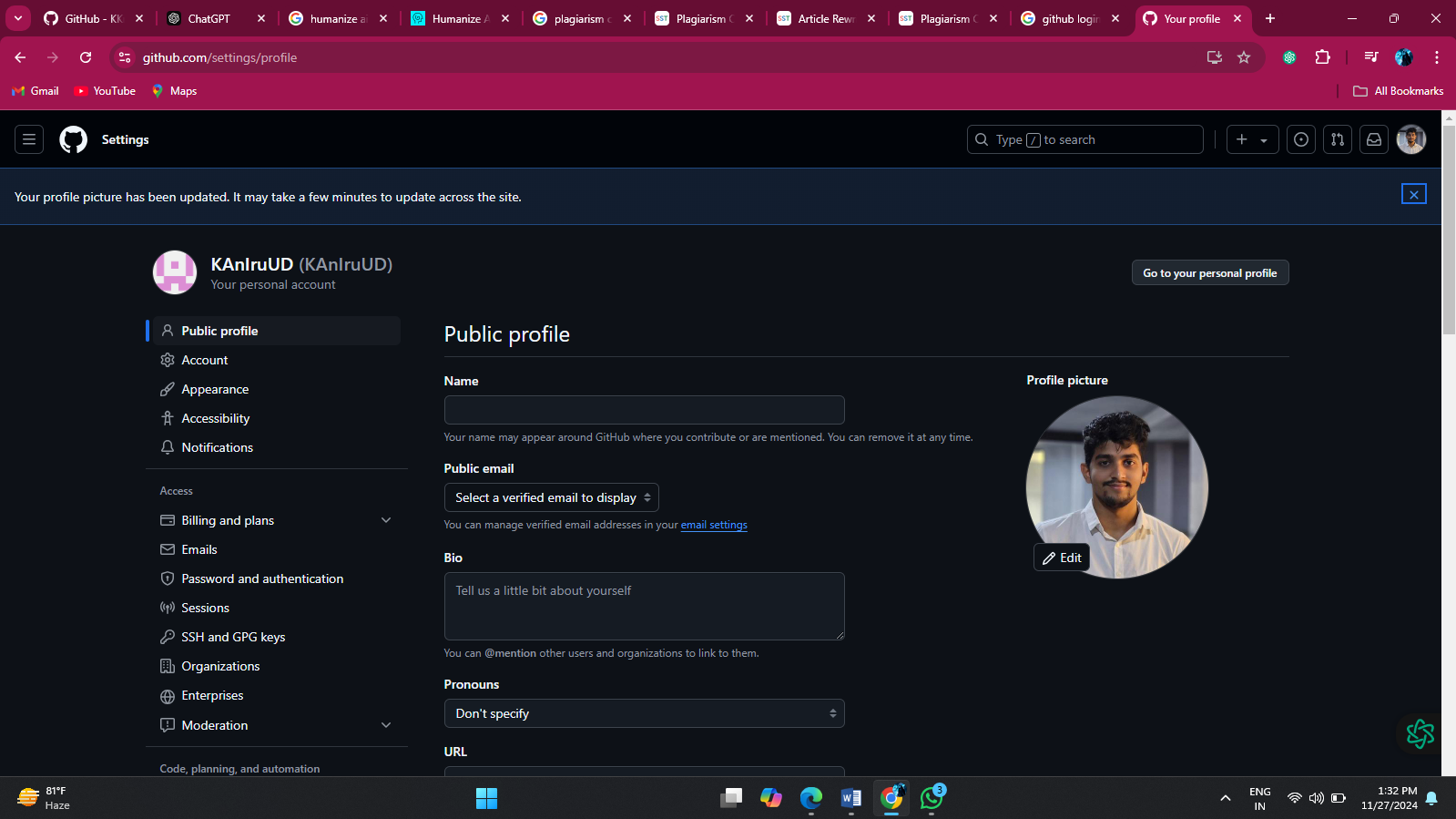
* GitHub will incite you to give a few data approximately your utilization of the benefit, e.g., individual ventures, examining, or collaboration with other groups.
* Fill in the discretionary areas and at that point press on "Total Setup".

**Step 7: Get to Dashboard**

After the setup of the account, you will be driven specifically to the GitHub dashboard from where you can make:

* Stores
* Browse open-source projects.
* Start joining up with other clients.





**How to Make a repository on GitHub**

**Step 1: Log in to GitHub**

* Open [GitHub](https://github.com/) in your browser.
* Log in utilizing your GitHub username and password.

**Step 2: Begin making a repository**

* Click the "+" symbol in the top-right corner of the GitHub dashboard.
* Select "Modern store" from the dropdown menu

**Step 3: Fill in the subtle elements of the repository**

**1. Store Title:** Enter a one of a kind title for your store (e.g., `MyFirstRepo`).

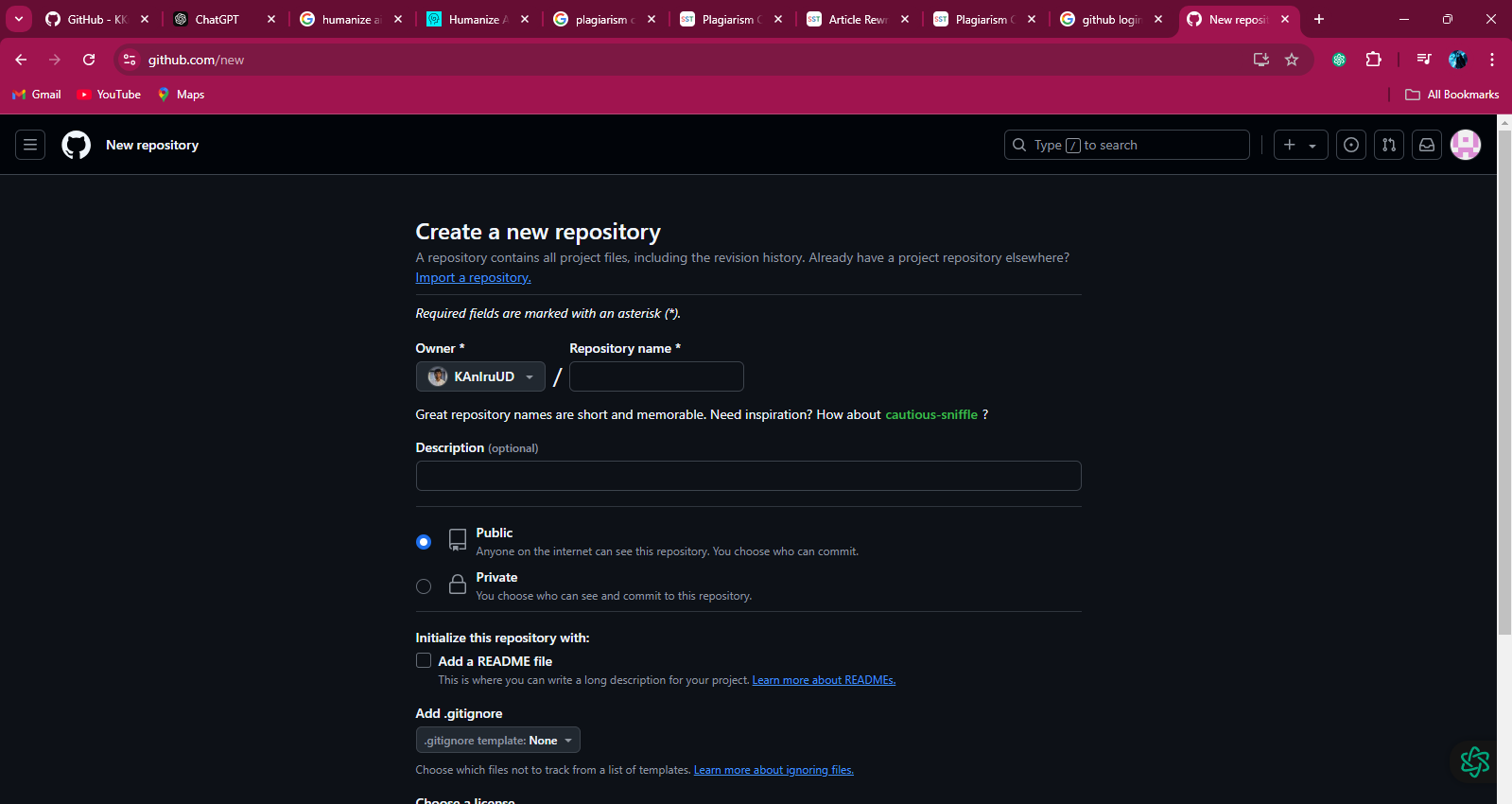
* This title will be portion of the store URL (e.g., `https://github.com/username/MyFirstRepo`).

**2. Depiction:** (Discretionary) Give a brief depiction of the extend.

* Case: "This is my to begin with GitHub store."

**3. Perceivability:**

* Open: Anybody can see the repository.
* Private: As it were you and collaborators can get to the store.



**Step 4: Initialize the Repository**

1. Select discretionary settings:

* Include a README record: Gives an outline of your venture.
* Include a .gitignore record: Prohibit records you don’t need to track (e.g., environment configurations).
* Include a permit: Indicate utilization authorizations for your extend.

2. These settings are discretionary and can be included afterward if needed.

**Step 5: Make the Repository**

* Press the "Make store" button at the foot of the shape.
* You’ll be diverted to the repository’s homepage.

**Step 6: Begin Utilizing the Repository**

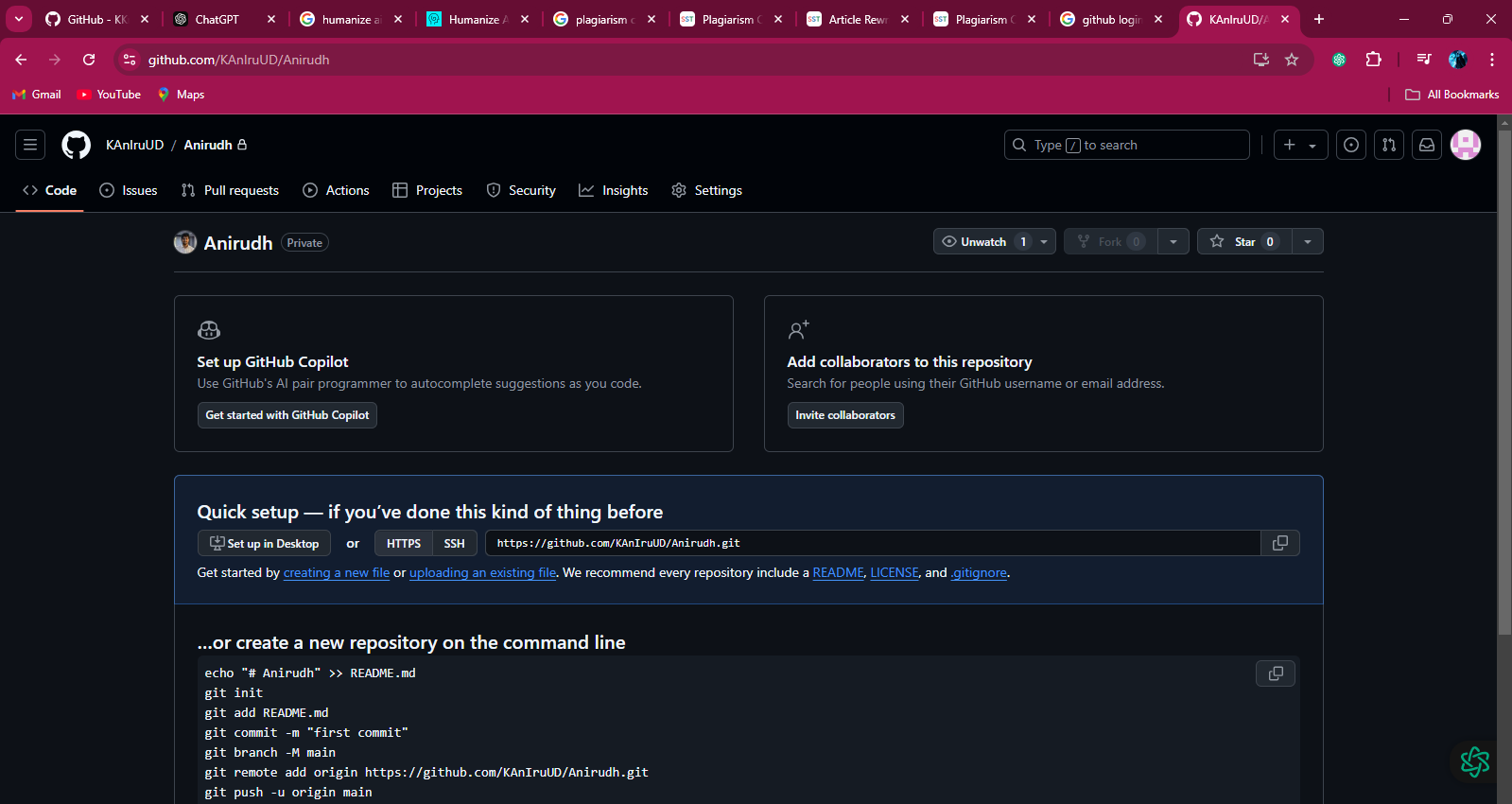
**1. Clone the Repository:**

* Duplicate the store URL and clone it to your neighborhood machine utilizing Git.
* Case command:```bash

git clone https://github.com/username/MyFirstRepo.git

**2. Transfer Records:**

* Include records specifically through the GitHub interface or from your nearby machine utilizing Git commands.

****