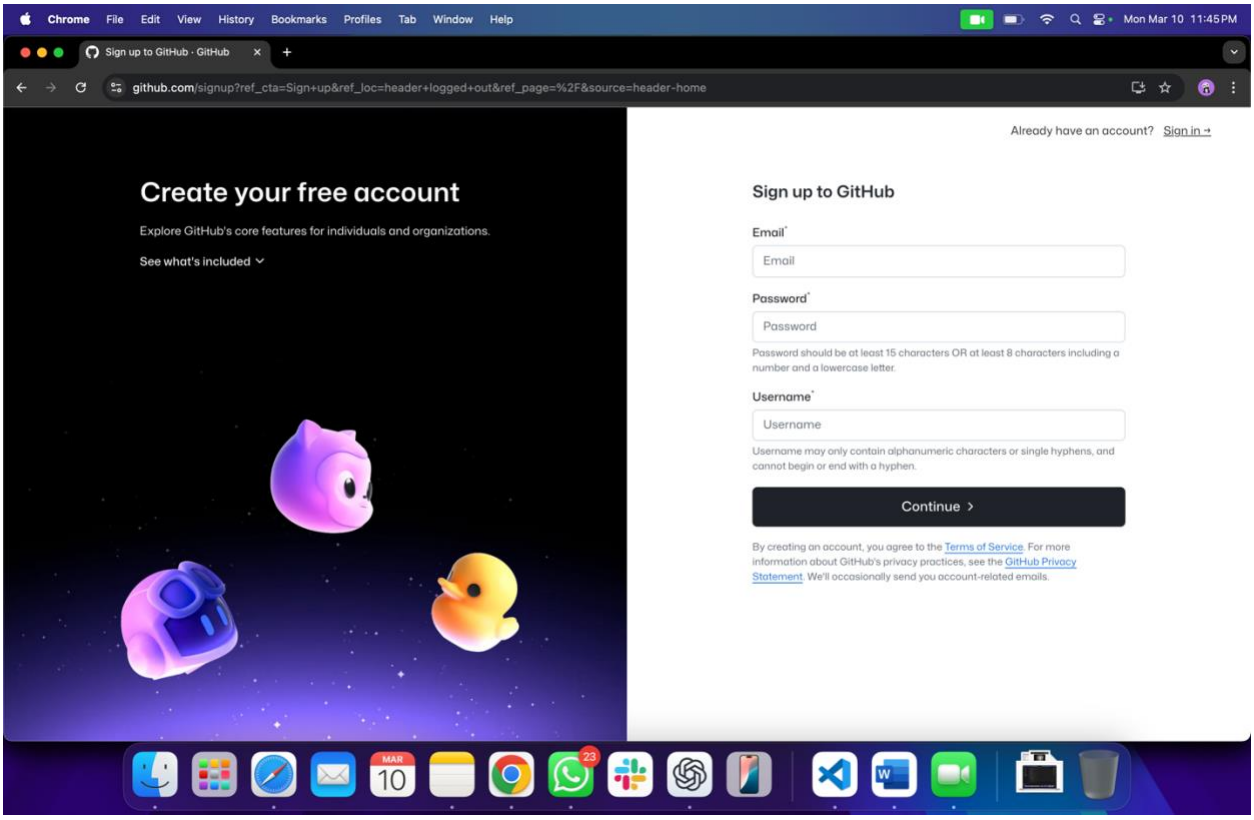
Getting Started with GitHub

1. Signing Up for GitHub

To begin using GitHub, visit GitHub and click on **Sign Up**.

- Sign up using your USC/Marshall email or a personal email account.
- Verify your account by entering the code sent to your registered email.



Note:

If you register with your school email, you can apply for the **GitHub Student Developer Pack**, which provides valuable benefits, including:

- **$200 in Digital Ocean credits** for cloud hosting.
- **$13/month Heroku credits** for 24 months to deploy your projects.
- **Free domain names** (e.g., .me, .live, .tech).
- **$50 MongoDB credits** for database management.

For a full list of benefits, visit: GitHub Student Developer Pack.

2. Activating the GitHub Student Developer Pack

To activate your **GitHub Student Developer Pack**:

1. Ensure you have signed up with your **USC email**.

2. Enable **two-factor authentication** (2FA) before applying:

   - Go to GitHub Security Settings.
   - Use an authentication app like **1Password, Authy, Microsoft Authenticator, or Google Authenticator**.
   - Scan the QR code and enter the generated key.
   - Download backup codes and store them safely.

3. Apply for the **Student Developer Pack** here.

4. Submit a copy of your **USC ID** and class schedule for verification.

5. Allow GitHub location access for identity confirmation.

6. Approval may take **3–10 business days** before benefits are activated.

3. Setting Up GitHub Desktop

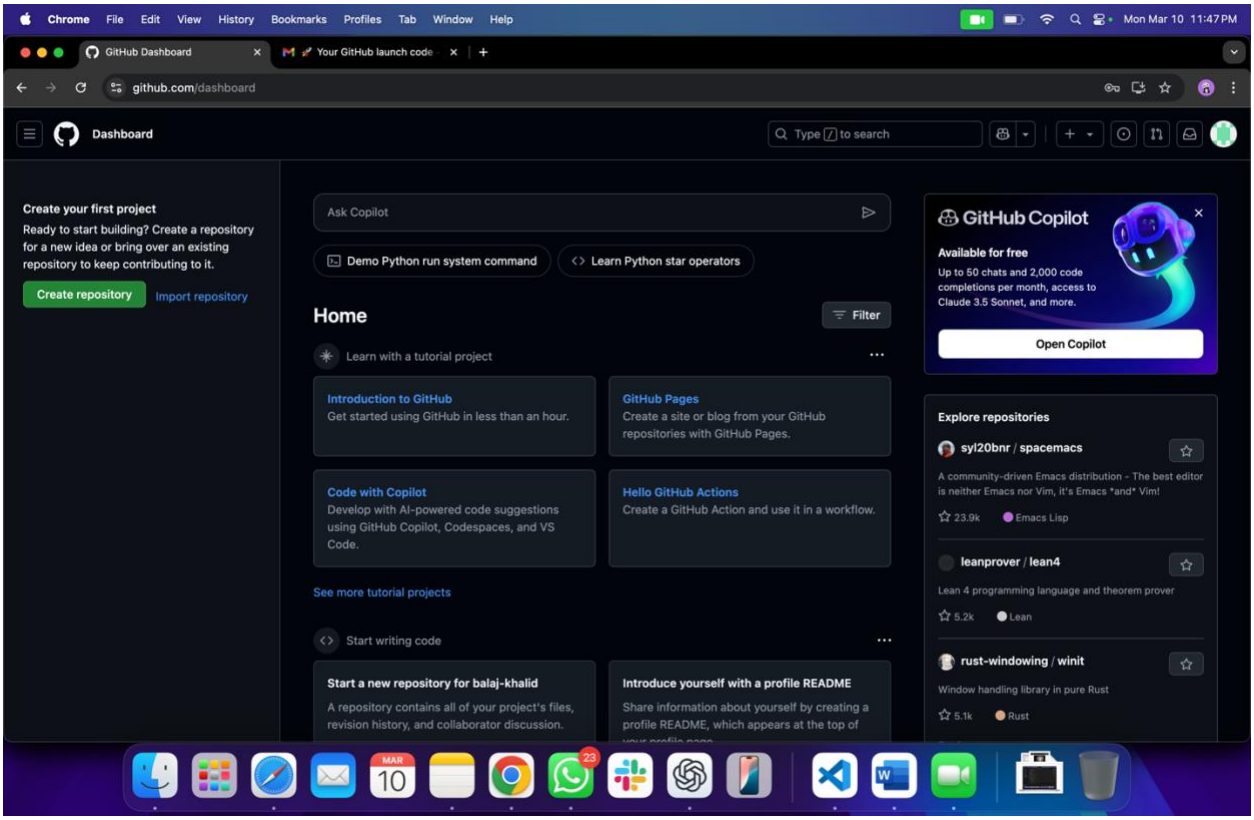
GitHub Desktop is a user-friendly application that simplifies managing repositories on your local machine.

Installing GitHub Desktop:

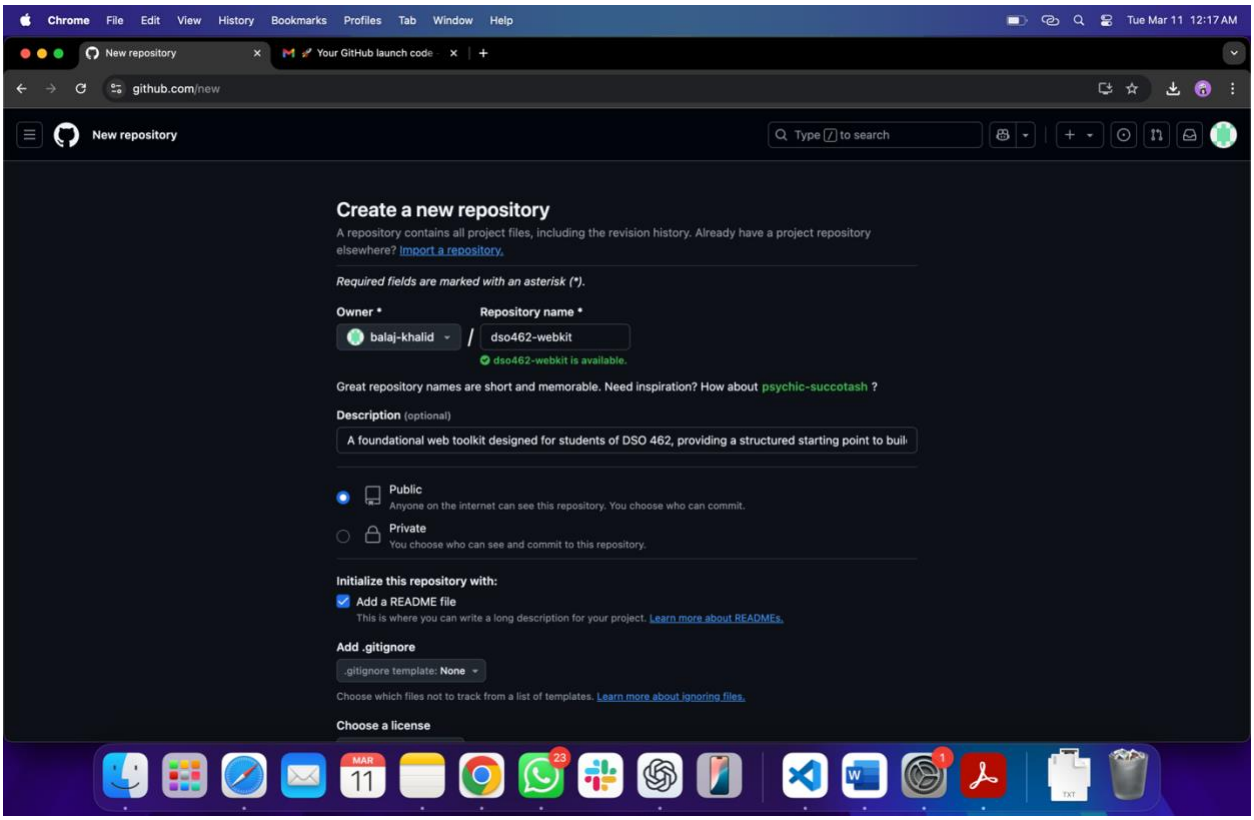
1. Download and install GitHub Desktop from desktop.github.com.
2. Sign in with your GitHub account.
3. Clone an existing repository or create a new one.
4. Use **GitHub Desktop** to commit changes and push updates effortlessly.

4. Creating Your First Repository

1. After signing in, navigate to your **GitHub homepage**.

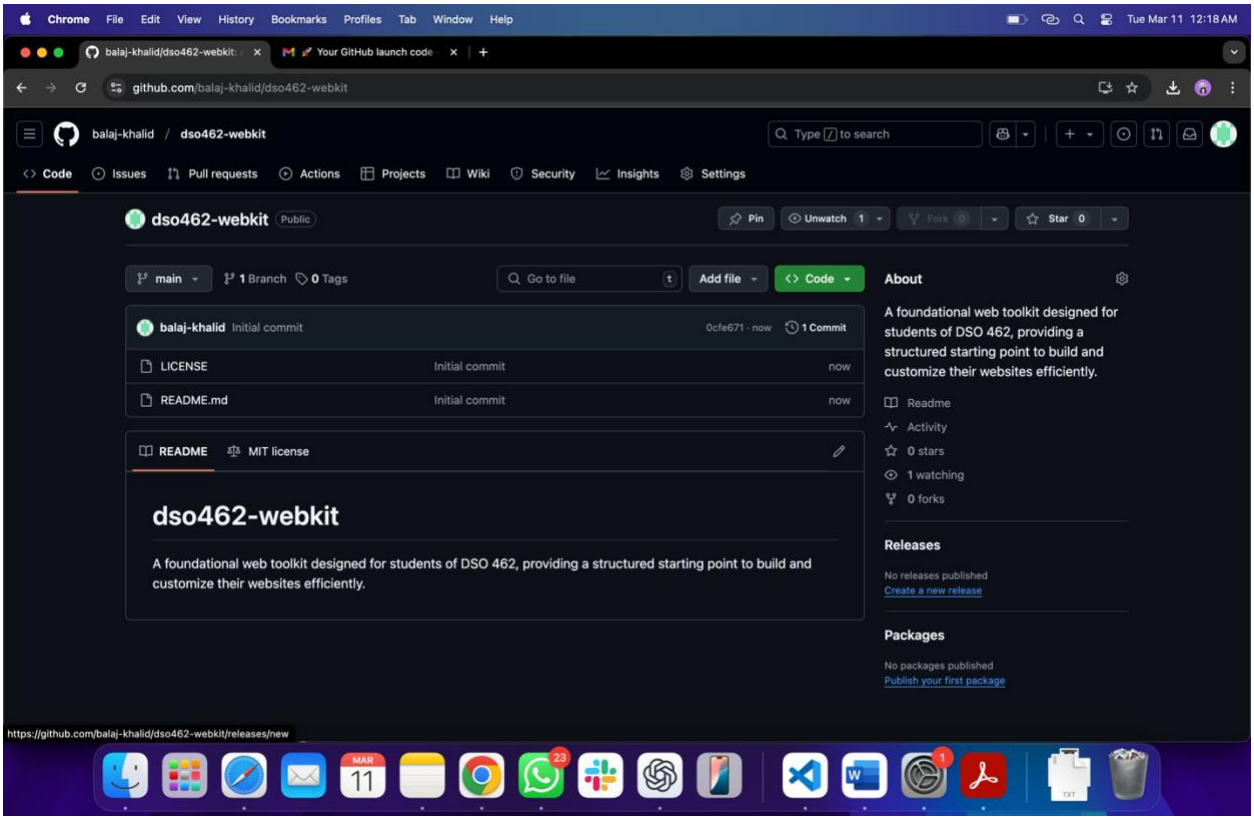


2. Click on **Create Repository** (left-hand panel).



3. Enter a unique name for your repository (e.g., project name, brand name).

4. Choose whether to make it public.
5. Click Create Repository.



6. Uploading Files to GitHub

There are multiple ways to upload files:

Using GitHub Web Interface:

1. Navigate to your repository.
2. Click **Add file** > **Upload files**.
3. Drag and drop files or select them manually.
4. Click **Commit changes** to save them.

Using GitHub Desktop:

1. Open GitHub Desktop and navigate to your repository.
2. Copy your project files into the repository folder.
3. Open GitHub Desktop and commit changes with a descriptive message.
4. Click **Push to GitHub** to upload your files.

Using Git (Command Line):

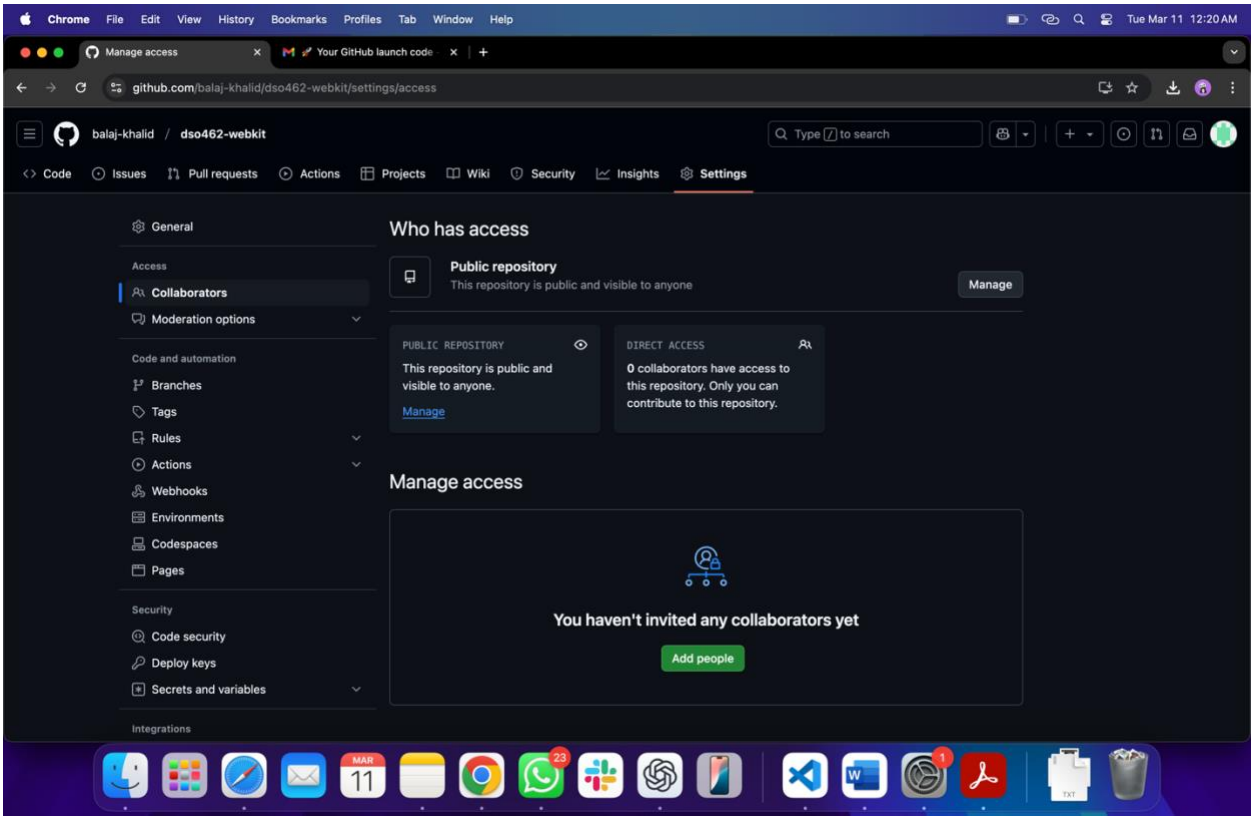
1. Open a terminal and navigate to your project folder.
2. Initialize Git (commands are encapsulated in ```):
   ``` git init ```
3. Add files to staging:
   ``` git add . ```
4. Commit changes (“Initial commit” is description of the commit):
   ``` git commit -m "Initial commit" ```
5. Link to GitHub repository:
   ```git remote add origin https://github.com/yourusername/repository-name.git```
6. Push files to GitHub:
   ``` git push -u origin main ```

6. Collaborating on GitHub

GitHub simplifies teamwork by enabling multiple contributors to work on the same project simultaneously.

Adding Collaborators:

1. Open your repository on GitHub.
2. Navigate to **Settings** > **Manage Access**.
3. Click **Invite Collaborator** and enter their **GitHub username or email**.
4. Assign appropriate access levels:
   a. **Read:** View repository but cannot make changes.
   b. **Write:** Make changes and push commits.
   c. **Admin:** Full control over repository settings.
5. Click **Send Invite** and wait for them to accept.
   ```



7. Benefits of Using GitHub for Collaboration

- **Version Control:** Track changes and revert to previous versions if needed.
- **Branching:** Work on features independently without affecting the main project.
- **Pull Requests:** Review code before merging it into the main branch.
- **Issue Tracking:** Manage bugs and tasks effectively.
- **Integration with CI/CD Tools:** Automate testing and deployment workflows.
- **Secure Hosting:** Store code safely with built-in security features.

By leveraging GitHub, you can **streamline development**, enhance **collaboration**, and manage projects **efficiently**. Happy coding!