**Assignment #03**

**Question No. 01:**

Git Hub:

1. History
2. Purpose
3. How to create an account
4. Terms and conditions
5. How to run git hub projects?

**Answer:**

**Git Hub:**

GitHub is a website that allows developers to host and share their code with others. It's often described as a "social coding" website because it allows developers to collaborate on projects, give feedback on code, and learn from each other. It's similar to other social media sites like Facebook or Twitter, but it's specifically designed for developers. GitHub also has a few unique features, like "pull requests," which allow developers to suggest changes to a project, and "repositories,"

1. **History:**

Certainly! Here's a more detailed history of GitHub:

1. \*2008:\*

- \*April 10:\* GitHub was officially launched by Tom Preston-Werner, Chris Wanstrath, and PJ Hyett.

- Initially focused on simplifying Git, the distributed version control system.

2. \*2011:\*

- GitHub gained significant popularity among developers and became a go-to platform for hosting open-source projects.

3. \*2012:\*

- \*July:\* GitHub reached a milestone with one million repositories hosted on the platform.

- GitHub secured $100 million in funding from Andreessen Horowitz.

4. \*2014:\*

- GitHub expanded its services beyond source code hosting, offering features like bug tracking, feature requests, and task management.

5. \*2018:\*

- \*June:\* Microsoft announced its acquisition of GitHub for $7.5 billion.

- \*October:\* The acquisition was completed, but GitHub continued to operate independently.

6. \*2019:\*

- GitHub introduced Actions, allowing developers to automate workflows directly on the platform.

7. \*2020:\*

- \*June:\* GitHub announced the initiative to rename the default branch from "master" to "main" to promote inclusivity and avoid language with historical overtones.

- GitHub continued to grow as a central hub for collaboration, hosting millions of developers and projects.

This timeline provides a snapshot of key events in GitHub's evolution, showcasing its journey from a Git repository hosting service to a comprehensive platform for collaborative software development.

**b) Purpose :**

The purpose of GitHub is to facilitate collaboration among software developers. It serves as a platform where developers can store, share, and work together on code. GitHub simplifies version control, enabling multiple people to contribute to a project without conflicts. It has become a central hub for open-source projects and a crucial tool for teamwork in the software development community.

**C) How to create an account:**

Creating a GitHub account is easy:

Visit GitHub: Go to the GitHub website in your web browser.

Sign Up: Click on the "Sign Up" button.

Fill in Details: Provide a username, email address, and password.

Verify Account: Complete any verification steps if prompted.

Choose Plan: Select a free plan unless you have specific needs.

Welcome to GitHub: Once completed, you're all set! You now have a GitHub account ready for coding adventures

**D) Terms and Conditions**:

GitHub's terms and conditions outline the rules for using their platform. Users agree:

* not to misuse the service,
* violate copyright,
* or engage in prohibited activities.
* GitHub has the right to suspend or terminate accounts that breach these terms
* . Users are responsible for their content and must comply with relevant laws.
* GitHub provides the service "as is" without warranties. It also highlights that changes to the terms may occur.
* Users should review the full terms on GitHub's website for detailed information

1. **How to run git hub projects?**

To run a GitHub project that uses Gradle, you can follow these steps:

1. \*Clone the Repository:\*

- Go to the GitHub repository of the project.

- Copy the repository URL.

- Open your terminal or command prompt.

- Use the git clone command followed by the repository URL to clone the project to your machine.

Example:

bash

git clone https://github.com/example/repository.git

2. \*Navigate to the Project Directory:\*

- Use the cd command to move into the directory of the cloned project.

Example:

bash

cd repository

3. \*Run Gradle Tasks:\*

- Gradle projects typically have a set of tasks defined in their build.gradle file.

- Use the ./gradlew (Unix-like systems) or gradlew.bat (Windows) script to run Gradle tasks.

Example:

bash

./gradlew build

4. \*Execute the Application:\*

- After running the necessary Gradle tasks, you can execute the application.

Example:

bash

./gradlew run

These are general steps, and the exact commands might vary depending on the project's structure and configuration. Always refer to the project's documentation or README file for specific instructions related to Gradle tasks and running the project

**Question No. 02:**

1. **How to upload projects on playstore?**

Uploading an Android app to the Google Play Store involves several steps. Here's a simplified guide:

1. \*Prepare Your App:\*

- Ensure your app is ready for release, has a unique package name, and is signed with a release key.

2. \*Create a Developer Account:\*

- Sign up for a Google Play Developer account at the Google Play Console website.

3. \*Pay the Registration Fee:\*

- Pay a one-time registration fee (as of my last knowledge update, it was $25).

4. \*Access Google Play Console:\*

- Log in to the Google Play Console using your developer account.

5. \*Create a New App:\*

- Click on "Create Application" and fill in details such as the default language, app title, and a short description.

6. \*Upload APK:\*

- Upload your signed APK (Android Package) to the Google Play Console under the "Release" tab.

7. \*Complete Store Listing:\*

- Provide detailed information about your app, including screenshots, a feature graphic, a comprehensive description, and other required assets.

8. \*Set Pricing and Distribution:\*

- Choose whether your app will be free or paid, and select the countries where it will be available.

9. \*Set Up In-App Products (if applicable):\*

- If your app includes in-app purchases, set up the necessary details in the "Monetize" section.

10. \*Define Content Rating:\*

- Complete the Content Rating questionnaire to determine the appropriate age group for your app.

11. \*Release Your App:\*

- Once all information is provided, click "Review" and then "Start Rollout to Production."

12. \*Opt-in to Google Play App Signing (optional but recommended):\*

- Google Play App Signing helps manage your app signing keys. You can opt in during the app creation process or later in the "Release" section.

13. \*Wait for Review:\*

- Google will review your app to ensure it complies with its policies. This can take a few hours to a few days.

14. \*Launch:\*

- Once approved, your app will be published on the Google Play Store.

**b- What is an .Apk file?**

An APK (Android Package) file is the installation package for Android apps. It contains the app's code, resources, manifest file, and necessary signatures, allowing users to download and install apps on their Android devices

**C-what is the latest price or fees to upload the projects on playstore?**

**-** Pay a one-time registration fee (as of my last knowledge update, it was $25) . which was 7037 rupees in Pakistani currency

**d-Is it possible to make a source code from .apk file? And which software converts the source code from apk file?**

Converting an APK back to source code involves decompilation, but it may violate terms of service or copyright. If you have legal access, tools like JADX or APKTool can be used for decompilation. Always respect intellectual property rights and legal considerations.

decompiling an APK and dealing with challenges like obfuscation and missing elements, is often done using tools like JADX, JADX-GUI, or JADX-Dex2Jar. Keep in mind the ethical and legal considerations when working with decompiled code.

**e- How much source code can be replicated and it may be usage for fullfledge a project?**

Decompiling an APK yields code, but it's not an exact copy. Obfuscation may complicate readability, and comments/formatting are often lost. Dynamic code might not fully translate, and third-party libraries may be absent. While the decompiled code can provide insights, it's usually insufficient for a full project. Moreover, using someone else's code without permission raises legal and ethical concerns. Always respect intellectual property rights and consider reaching out to the original developers for collaboration or permission if needed.