

**Basics of Open Source Demystified**

Project Report submitted in partial fulfilment of the requirements for the

award of the degree of

BACHELOR OF COMPUTER APPLICATIONS (BCA)

Department of Computer Science (UG)



Submitted by

# Alex E J

# 22BCAC09

Under the guidance of

# Prof. Margaret Mary

**Department of Computer Science (UG) BCA Programme**

**Kristu Jayanti College (Autonomous)**

**K. Narayanapura, Kothanur P.O., Bangalore – 560077**



**DEPARTMENT OF COMPUTER SCIENCE (UG)**

# CERTIFICATE OF COMPLETION

This is to certify that the project entitled **“Basics of Open Source Demystified”** has been satisfactorily completed by **(name), 22BCAC09** in partial fulfilment of the award of the Bachelor of Computer Applications degree requirements prescribed by Kristu Jayanti College (Autonomous) Bengaluru (Affiliated to Bangalore North University) during the academic year 2023-2024.

***Internal Guide Head of the Department***

***Valued by Examiners***

**1:**  ***Centre:* Kristu Jayanti College**

**2:**  ***Date:***



**DECLARATION**

I **Alex E J, 22BCAC09** hereby declare that the project work entitled **“Basics of Open Source Demystified”** is an original project work carried out by me, under the guidance of **Prof. Margaret Mary**

This project work has not been submitted earlier either to any University / Institution or any other body for the fulfilment of the requirement of a course of study.

Signature

**Alex E J**

Bengaluru Date:

**ACKNOWLEDGEMENT**

*“The success is the result of preparation, hard work, and learning from failure*.”

It’s my duty to acknowledge and thank the individuals who has contributed to the successful completion of the project.

I take this opportunity to express my profound and wholehearted thanks to **Rev**. **Fr. Dr. Augustine George, Principal** and **Rev. Fr. Lijo P Thomas**, **Vice Principal** and **Chief Financial Officer, Kristu Jayanti College, Bangalore**.

I am deeply grateful for the invaluable support and encouragement provided by **Dean Dr. Calistus Jude AL** throughout this journey. His steadfast guidance has been instrumental in shaping my academic path, and sincerely appreciative of his contributions to my development . I also extend my heartfelt thanks our beloved and resourceful

**Head of the Department , Prof. Sevuga Pandian A, and Programme Coordinator ,**

**Dr. Sasikumar V R,** for their invaluable guidance and support. Additionally, I wish to express my deep gratitude to **Prof.Margaret Mary**, my mentor and guide, whose mentorship and profound insights have greatly contributed to my growth and learning journey. Your willingness to share expertise and offer constructive feedback has enriched my learning experience immensely.

It is my duty to express my thanks to all Teaching and Non-Teaching Staff members of Computer science department who offered me help directly or indirectly by their suggestions.

The successful completion of my project would not have been possible without my parent’s Sacrifice, guidance, and prayers. I take this opportunity to thank everyone for their continuous Encouragement. I convey my thankfulness to all my friends who were with me to share my happiness and agony.

Last but not the least I thank almighty God for giving me strength and good health throughout

my project and enabling me to complete it successfully

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Topics** | **Page.No** |
| **1** | **Lab Procedures** |  |
| 1.1 | Git Commands | 1-2 |
| 1.2 | Cloning A Repository | 3-4 |
| 1.3 | Forking A Repository | 5 |
| 1.4 | Performing The First Commit | 6 |
| 1.5 | Push Changes to Repository | 7-8 |
| 1.6 | Pull Changes from Repository | 9 |
| 1.7 | Credential Manager | 10 |
| 1.8 | Creating A New Issue | 11 |
| 1.9 | Pull Request | 12-14 |
| 1.10 | Perform Merge Conflict | 15 |
| 1.11 | Branching | 16 |
| 1.12 | Creating Personal Access, Token | 17-18 |
| **2** | **Project Works** | 29-29 |
| 2.1 | AOSP | 29-24 |
| 2.2 | Visual Studio Code | 25-29 |
| **3** | **How to Join and Contribute to Open Source** | 34-37 |
| **4** | **Community Meeting Minutes** | 30 - 33 |
| 4.1 | Meeting Report | 30-31 |
| 4.2 | Tech Talk Report | 32-33 |

1. **Lab Procedures**

### Procedure 1: Git Commands

Clone - Clone a repository into a new directory

Init - Create an empty Git repository or reinitialize an existing one Add - Add file contents to the index

Mv - Move or rename a file, a directory, or a symlink Restore - Restore working tree and from the index

Bisect - Use binary search to find the commit that introduced a bug Diff - Show changes between commits, commit and working tree,etc Grep - Print lines matching a pattern

Log - Show content logs

Show - Show various types of objects Status - Show the working tree status Branch - List, create or delete branches Commit - Record changes to the repository

Merge - Join two or more development histories together Rebase - Reapply commits on top of another base tip Switch - Switch branches

Tag - Create, list, delete or verify a tag object signed with GPG Fetch - Download objects and refs from another repository

Pull - Fetch from and integrate with another repository or a local branch Push - Update remote refs along with associated objects

### Procedure 2: Cloning A Repository

Step 1: Install Git

If you don't have Git installed, you can download and install it from the official website: Git Downloads

Step 2: Open a Terminal or Command Prompt or powershell

Open your terminal or command prompt. This could be Git Bash, Command Prompt (Windows), or Terminal (Mac/Linux).

Step 3: Navigate to the Desired Directory

Navigate to the directory on your local disk where you want to clone the repository. You can use the cd command to change directories. For example:

cd path/to/your/directory

Step 4: Get the Repository URL

Go to the GitHub repository you want to clone and copy the repository URL. You can find it on the repository's page, usually under the "Code" button.

Step 5: Clone the Repository

Use the git clone command to clone the repository. Replace <repository\_url> with the actual URL you copied.

git clone <repository\_url>

For example:

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

Step 6: Enter Credentials (if required)

If the repository is private, Git may prompt you to enter your GitHub username and password or personal access token.

Step 7: Verify the Cloning Process

Git will download the repository to your local disk. Once the process is complete, you should see a new directory with the repository name in your specified local directory.

Step 8: Navigate into the Cloned Repository

Navigate into the cloned repository directory using the cd command: cd repository

### Procedure 3: Forking A Repository

1. Navigate to the Repository: Go to the repository you want to fork on the Git hosting platform (e.g., GitHub).
2. Click the “Fork” Button: Look for the “Fork” button usually located in the upper right corner of the repository page. Clicking this button will initiate the forking process.

1. Choose Where to Fork: If you have multiple accounts or organizations, you might be prompted to choose where to fork the repository.
2. Wait for Forking to Complete: The platform will create a copy of the repository under your account. This process may take a few moments, depending on the size of the repository.
3. Clone Your Fork: Once the forking process is complete, clone your forked repository to your local machine using the git clone command. Replace <repository\_URL> with the URL of your forked repository: git clone <repository\_URL>
4. Make Changes: Now you can make changes to the code, add new features, fix bugs, etc., in your local copy of the repository.

### Procedure 4: Performing The First Commit

1. Initialize a Git Repository: If you haven’t already done so, navigate to the root directory of your project in the terminal and initialize a new Git repository using the following command: git init
2. Add Files to the Staging Area: Use the git add command to add files or directories to the staging area. For example, to add all files in the current directory and its subdirectories, you can use: git add .
3. Commit the Changes: Once you’ve added the files to the staging area, you can commit them to the repository using the git commit command. This command will open a text editor (usually Vim or Nano) for you to enter a commit message. git commit
4. Alternatively, you can provide a commit message directly on the command line using the - m flag: git commit -m "Initial commit"
5. Verify the Commit: After committing, you can use git log to view the commit history and verify that your first commit was successful: git log

### Procedure 5: Push Changes to Repository

To push changes to a repository on GitHub using a personal access token (PAT), follow these steps:

Commit Changes: Make the necessary changes to your files in your local repository.

Stage Changes: Use the git add command to stage the changes you want to include in the commit. For example:

git add .

Commit Changes: Commit the staged changes with a descriptive message using the git commit command. For example:

git commit -m "Your commit message here"

Push Changes: Use the git push command to push your commits to the remote repository on GitHub. Include your PAT in the URL as a password. The syntax is similar to the clone command but with git push instead of git clone. For example:

git push https://<your-username>:<your-token>@github.com/<your- username>/<repository>.git

Replace <your-username> with your GitHub username, <your-token> with the token you generated, and <repository> with the name of your repository.

Enter Credentials: You may be prompted to enter your GitHub username and the PAT you generated. Enter them when prompted.

Verify Changes: Once the push is successful, visit your repository on GitHub to verify that the changes have been pushed successfully.

Important Notes:

Treat your personal access token like a password. Keep it secure and do not share it publicly.

It's recommended to use HTTPS for pushing changes to repositories when using a personal access token.

If you're pushing to a repository where you have write access, your changes should now be reflected in the remote repository on GitHub.

### Procedure 6: Pull Changes from Repository

Make Changes in Remote Repository (Optional): If you want to make changes directly in the remote repository (e.g., via a web interface or another user's contribution), do so.

Navigate to Your Repository Directory: Use the cd command to navigate to the directory of your local repository: cd C:\path\to\your\repository

Pull Changes from Remote Repository: Use the git pull command to fetch and merge changes from the remote repository into your local repository. Specify the remote repository's name (typically origin) and the branch name (e.g., master):

git pull origin master

If you're pulling from a branch other than master, replace master with the appropriate branch name.

Authentication (if prompted): If authentication is required, provide your credentials.

Verify Updated Code: After the pull operation completes, verify that the changes made directly in the remote repository are now reflected in your local repository.

### Procedure 7: Credential Manager

To remove a GitHub account from Credential Manager on a Windows machine, you can follow these steps:

Open Credential Manager: Press Win + S to open the search bar. Type "Credential Manager" and select it from the search results.

Locate GitHub credentials: In Credential Manager, you'll see a list of stored credentials. Look for any entries related to GitHub.

Remove GitHub credentials: Click on the entry for GitHub, and then click on the "Remove" or "Delete" option. This will delete the stored credentials for that GitHub account.

Confirm removal: Confirm that you want to remove the credentials if prompted.

### Procedure 8: Creating A New Issue

Creating an Issue

1. Navigate to Repository: Go to the repository on the Git hosting platform (e.g., GitHub).
2. Click on “Issues”: Look for the “Issues” tab and click on it to access the issues section of the repository.
3. Create New Issue: Click on the “New Issue” button to create a new issue.
4. Provide Details: Enter a title and description for the issue, including relevant information such as what needs to be addressed, steps to reproduce the problem, and any other pertinent details.
5. Assignees, Labels, and Milestones: Optionally, you can assign the issue to specific individuals, apply labels to categorize it, and set milestones to track progress.
6. Submit Issue: Once you’ve filled in the details, click on the “Submit new issue” button to create the issue.

#### Templates for ISSUE

\*\*Issue/Feature Description:\*\*

\*\*Why this issue to fixed / feature is needed(give scenarios or use cases):\*\*

\*\*How to reproduce, in case of a bug:\*\*

\*\*Other Notes / Environment Information: (Please give the env information, log link or any useful information for this issue)\*\*

### Procedure 9: Pull Request

Creating a Pull Request

1. Fork the Repository: If you haven’t already, fork the repository you want to contribute to. This creates a copy of the repository under your account.
2. Clone Your Fork: Clone your forked repository to your local machine using the git clone command.

git clone <repository\_URL>

1. Create a Branch: Create a new branch to work on your changes. It’s good practice to create a separate branch for each feature or issue you’re addressing.

git checkout -b <branch\_name>

1. Make Changes: Make the necessary changes to the codebase on your local branch.
2. Commit Changes: Once you’ve made your changes, commit them to your local repository with descriptive commit messages.

git add .

git commit -m "Description of changes"

1. Push Changes: Push your local branch to your forked repository on the Git hosting platform. git push origin <branch\_name>
2. Create Pull Request: Navigate to your forked repository on the Git hosting platform and click on the “Pull Request” button.
3. Select Branches: Choose the branch you pushed your changes to in your fork as the “compare” branch, and the original repository’s branch you want to merge into as the “base” branch.
4. Review Changes: Provide a title and description for your pull request, summarizing the changes you made. You can also include any relevant context or details.
5. Submit Pull Request: Once you’re satisfied with the changes and everything looks good, submit the pull request.
6. Review and Merge: The maintainers of the original repository will review your pull request, provide feedback if necessary, and merge it into the main codebase if approved.

#### Template For Pull Request

1. Please give clear description and fill all the needed fields in the PR template below
2. Provide all the test report and results for the PR. It is mandatory. Otherwise, your PR may get rejected without any review/discussion
3. If the PR is incomplete/in progress, please add [WIP] at the beginning of the PR title.
4. Provide the link to the issue and other relevant files related to the PR

-->

\*\*What type of PR is this?\*\*

* + Uncomment only one ` /kind <>` line, hit enter to put that in a new line, and remove leading whitespace from that line:

>

* + /kind new feature
  + /kind bug fix
  + /kind cleanup
  + /kind revert change
  + /kind design
  + /kind documentation
  + /kind enhancement

\*\*What this PR does / why we need it\*\*:

\*\*Which issue(s) this PR fixes\*\*:

<!--

\*Please provide the issues number or link.

Usage: `Fixes #<issue number>`, or `Fixes (paste link of issue)`.

-->

Fixes #

\*\*Test Report Added?\*\*:

* + Uncomment only one ` /kind <>` line, hit enter to put that in a new line, and remove leading whitespace from that line:
  + /kind TESTED
  + /kind NOT-TESTED

\*\*Test Report\*\*:

<!--

\*Please provide the test report link (public accessible, screen shot or copy paste the test report, or add the testing details.

### Procedure 10: Perform Merge Conflict

Pull the Latest Changes: Before making any changes, ensure your local repository is up-to- date with the remote repository. Pull the latest changes from the remote branch you intend to merge into.

git checkout <your\_branch>

git pull origin <remote\_branch>

Merge the Target Branch: Merge the target branch into your current branch. git merge <target\_branch>

Resolve Merge Conflicts: If there are conflicting changes, Git will notify you about the conflicts and mark them in the affected files. You need to manually resolve these conflicts.

Open Files with Conflicts: Open the conflicted files in your preferred text editor. Git marks the conflicting sections within the file using special markers like <<<<<<<, =======, and >>>>>>>.

Resolve Conflicts Manually: Inside the conflicted file(s), locate the conflicting sections marked by Git and decide how to resolve them. You can choose to keep one version of the code, combine both versions, or write an entirely new version. Remove the conflict markers (<<<<<<<, =======, >>>>>>>) once you've resolved the conflicts.

Stage the Changes: After resolving the conflicts, stage the changes using the git add command. git add <conflicted\_file(s)>

Commit the Changes: Once you've resolved all conflicts in the files, commit the changes. git commit -m "Resolved merge conflicts"

### Procedure 11: Branching

**Creating a Branch**: You can create a new branch directly on the GitHub website or using Git commands locally on your machine (git checkout -b branch-name).

**Making Changes**: After creating a branch, you can make changes to the codebase, add new files, modify existing ones, etc.

**Committing Changes**: Once you've made changes, you commit them to the branch using git commit with a commit message describing the changes.

**Pushing Changes**: If you're working locally, you push your branch to the GitHub repository using git push origin branch-name.

**Pull Request**: When you're ready to merge your changes back into the main codebase, you create a pull request (PR) on GitHub. This allows others to review your changes before merging.

**Review and Merge**: Other contributors review your changes in the pull request. Once approved, the changes are merged into the main branch.

**Deleting Branch**: After merging, you can delete the branch both locally (git branch -d branch-name) and on GitHub, to keep your repository clean.

### Procedure 12: Creating Personal Access, Token

Git clone https://<your-username>:<your-token>@github.com/<your- username>/<repository>.git

To create a personal access token (PAT) on GitHub, you can follow these steps: Log in to GitHub: Make sure you are logged in to your GitHub account.

Access Settings: In the upper-right corner of GitHub, click on your profile picture, and then click on "Settings."

Navigate to Developer Settings: In the left sidebar, click on "Developer settings," and then click on "Personal access tokens."

Generate a New Token: Click on the "Generate token" button.

Configure Token Settings: Enter a name for your token in the "Token name" field.

Select the scopes (permissions) that your token should have. Be cautious and only grant the necessary permissions.

Optionally, set an expiration period for the token.

Generate Token: Click the "Generate token" button at the bottom.

Copy the Token: GitHub will generate the token. Copy the token immediately as you won't be able to see it again.

Important Notes: Treat your personal access token like a password. Keep it secure and do not share it publicly.

If you need to use the token in a command-line operation (e.g., for Git operations), include it in the URL as a password. For example: git clonehttps://<your-username>:<your- token>@github.com/<your-username>/<repository>.git

Replace <your-username> with your GitHub username and <your-token> with the token you generated.

## 2.1 Android Open Source Project (AOSP)

## A pop of color and more: updates to Android's brand

### Introduction

The Android Open Source Project (AOSP) is a collaborative effort that serves as the foundation for the Android mobile operating system. Managed by the Open Handset Alliance (OHA), AOSP embodies the principles of openness, transparency, and community-driven development

### Project Summary

|  |  |
| --- | --- |
| Website | https://source.android.com/ |
| Organisation/Foundation name | Open Handset Alliance (OHA) |
| Licence | Apache-2.0 licence |
| Open/Proprietary | Open Source |
| Source Path | https://android.googlesource.com |
| Brief Description | AOSP represents a monumental open-source initiative aimed at providing a robust and versatile platform for mobile devices. Rooted in the collaborative efforts of the global developer community, it offers a comprehensive suite of tools, frameworks, and resources for building innovative mobile applications. |

**Project Details**

**Key features:**

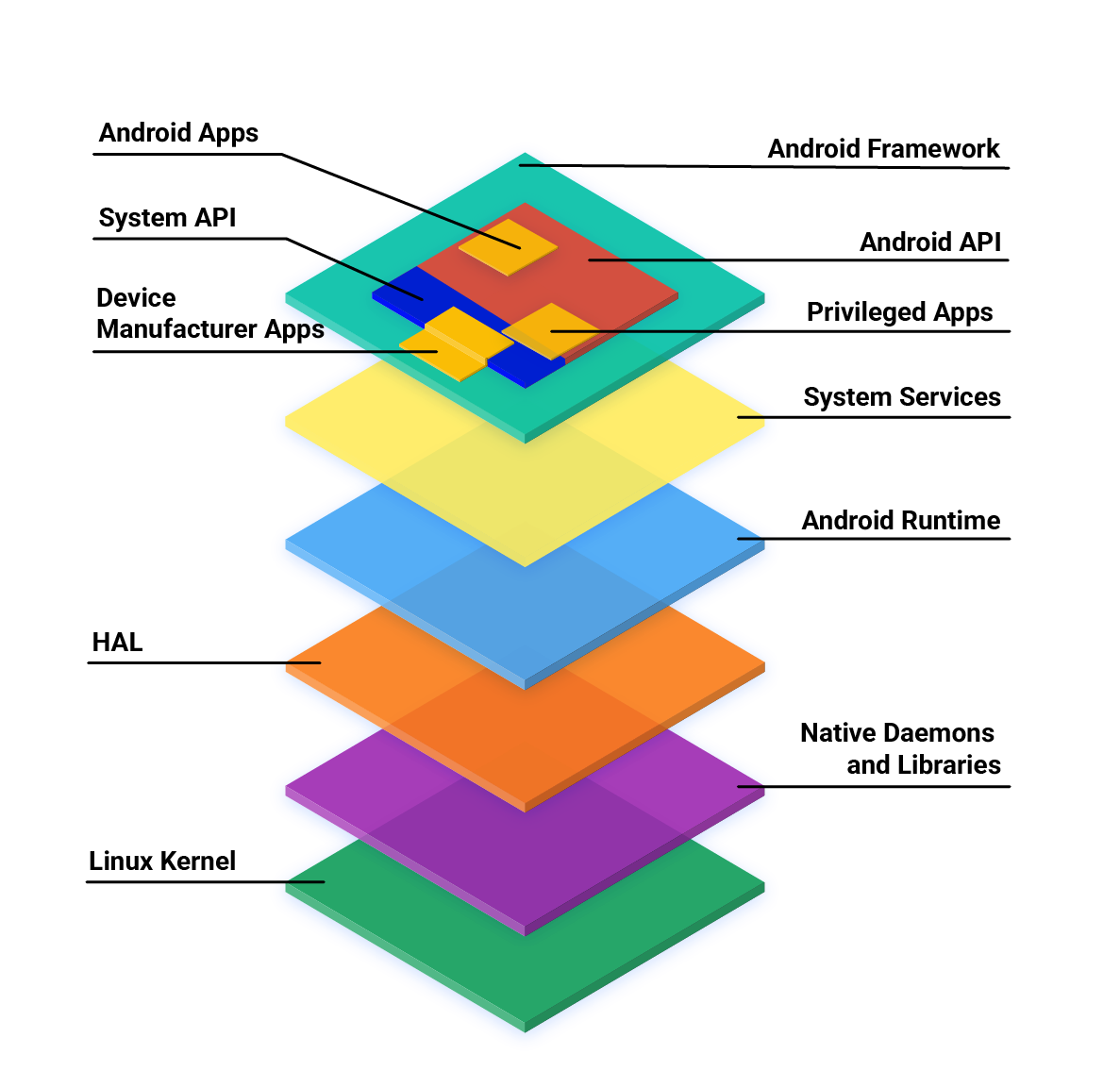
**1. Android Runtime (ART):** Android Runtime (ART) is the managed runtime environment used by the Android operating system. It replaces the older Dalvik virtual machine and offers improved performance, memory management, and application responsiveness. ART employs ahead-of-time (AOT) compilation, which translates bytecode into native machine code during app installation, resulting in faster execution and reduced memory footprint.

**2.** **Extensive Application Framework:** The Android Application Framework provides developers with a comprehensive set of APIs and libraries for building diverse and feature-rich applications. This framework includes components for user interface design, data storage, multimedia playback, network communication, and more. By leveraging these APIs, developers can create engaging and interactive applications that leverage the full capabilities of the Android platform.

**3.Customizability:** AOSP offers extensive customizability options, allowing device manufacturers, developers, and users to tailor the Android experience to their specific preferences and requirements. From custom user interfaces and themes to device-specific optimizations and configurations, AOSP provides the flexibility needed to create unique and differentiated Android experiences. This customizability extends to the development of custom ROMs, where community-driven efforts often introduce innovative features and enhancements that cater to niche user demographics and use cases.

**4.Security Features:** Security is a top priority for the Android platform, and AOSP incorporates a variety of built-in security mechanisms and protocols to safeguard user data and privacy. These features include sandboxing of applications to prevent unauthorized access to system resources, secure inter-process communication (IPC) mechanisms, encryption of sensitive data, and robust authentication and authorization frameworks. Additionally, Google regularly releases security updates and patches to address vulnerabilities and mitigate emerging threats.

**Architecture**



AOSP consists of several essential components:

**1**. **AOSP Core Framework:**

The core framework of AOSP serves as the foundation for the Android operating system. It includes essential components such as the Android Runtime (ART), system services, and system libraries. These components provide the underlying infrastructure for running Android applications and managing system resources.

* **Android Runtime (ART):** The runtime environment responsible for executing Android applications. ART utilizes ahead-of-time (AOT) compilation to translate bytecode into native machine code, enhancing performance and reducing startup time.
* **System Services:** Essential background services that manage system-level functionalities such as network connectivity, power management, and resource allocation.
* **System Libraries:** Shared libraries that provide common functionality to Android applications, including graphics rendering, database access, and encryption algorithms.

**2**. **Linux Kernel:**

The Linux kernel forms the core of the Android operating system, providing hardware abstraction, device drivers, and essential system services. It facilitates communication between hardware components and higher-level software layers, ensuring seamless operation across diverse hardware platforms.

* **Hardware Abstraction:** The kernel abstracts hardware-specific details and provides a unified interface for interacting with hardware components, such as processors, memory, storage, and input/output devices.
* **Device Drivers:** Kernel modules that facilitate communication between the operating system and hardware peripherals, ensuring compatibility and functionality across a wide range of devices.
* **System Services:** Core kernel services responsible for process management, memory management, file system handling, and device input/output operations.

**3**. **Application Framework:**

The Android Application Framework comprises a rich set of APIs and libraries that enable developers to build interactive and feature-rich applications for Android devices. This framework provides components for user interface design, multimedia playback, data storage, network communication, and more, empowering developers to create diverse and engaging user experiences.

* + - **User Interface Toolkit:** A collection of APIs and widgets for creating intuitive and visually appealing user interfaces, supporting layouts, views, animations, and themes.
    - **Content Providers:** Abstraction layer for accessing and manipulating data stored on the device or shared between applications, facilitating seamless data sharing and integration.
    - **Notification Manager:** System service for managing notifications and alerts, providing users with timely updates and information from installed applications.

1. **Security Features:**

Security is a paramount concern in AOSP, with multiple layers of protection implemented to safeguard user data and privacy. Important security features include:

* + - **Application Sandboxing:** Isolating applications from each other and the underlying system to prevent unauthorized access to sensitive resources and data.
    - **Permissions Framework:** Granting applications access to specific device features and data based on user consent and predefined permissions, ensuring fine-grained control over app behavior.
    - **Secure Boot:** Verifying the integrity of the boot process and system software to prevent tampering and unauthorized modifications, enhancing the overall security posture of the device.

1. **Device Abstraction Layer (HAL):**

The Device Abstraction Layer (HAL) provides a standardized interface for interacting with hardware components on Android devices. This layer abstracts hardware-specific details and exposes a uniform API for device manufacturers and developers, facilitating hardware compatibility and interoperability across different device models and manufacturers.

1. **Updates and Enhancements:**

AOSP undergoes continuous development and improvement, with updates and enhancements introduced regularly to address security vulnerabilities, enhance system stability, and introduce new features. Updates are typically delivered over-the-air (OTA) or through manual installation, ensuring that Android devices remain up-to-date with the latest advancements and user expectations.

1. **Community Contributions:**

AOSP benefits from contributions from a diverse and vibrant community of developers, device manufacturers, and enthusiasts. Community contributions encompass code contributions, bug fixes, feature requests, and documentation updates, driving the evolution and innovation of the Android platform. Community involvement fosters collaboration, knowledge sharing, and collective problem-solving, ensuring the continued success and relevance of AOSP in the ever-changing landscape of mobile technology.

Top of Form

### Current Usage

A list of notable organizations and products known to be using the Android Open Source Project (AOSP) as of the last update:

**1.** **Google**: Google, the organization behind the development of AOSP, utilizes it as the foundation for the Android operating system across a wide range of devices, including smartphones, tablets, wearables, and automotive systems. Google integrates AOSP with proprietary components and services to deliver the Android experience to users worldwide.

**2. Samsung**: Samsung, a leading manufacturer of Android devices, relies on AOSP as the basis for its custom Android firmware. Samsung's Galaxy series of smartphones and tablets run on modified versions of AOSP, incorporating additional features, optimizations, and enhancements tailored to Samsung's hardware and software ecosystem.

**3. Huawei**: Huawei, another prominent Android device manufacturer, leverages AOSP to develop its EMUI (Emotion UI) custom Android skin. EMUI powers Huawei's smartphones and tablets, offering a unique user experience with advanced features and optimizations designed to complement Huawei's hardware lineup.

**4. Xiaomi**: Xiaomi, known for its affordable yet feature-rich Android devices, uses AOSP as the foundation for its MIUI (MI User Interface) custom Android ROM. MIUI adds a layer of customization and personalization to AOSP, offering Xiaomi users a highly customizable and user-friendly software experience.

**5. Custom ROM Communities:** Various custom ROM communities, such as XDA Developers and Android Open Kang Project (AOKP), utilize AOSP as the basis for their custom Android firmware. These communities provide enthusiasts and power users with the opportunity to customize and enhance their Android devices with unique features, performance optimizations, and the latest Android updates beyond what is offered by device manufacturers.

## Technical Details

1. **Scalability**:

* AOSP is designed to be highly scalable, capable of supporting a vast ecosystem of Android devices ranging from smartphones and tablets to wearables, automotive systems, and IoT devices.
* Its modular architecture allows for the efficient utilization of system resources and supports horizontal scaling, enabling organizations to deploy AOSP across large-scale device fleets and diverse hardware configurations.
* AOSP's flexible deployment options, including custom ROMs and device-specific builds, accommodate the scalability requirements of device manufacturers and developers seeking to deliver Android-based solutions at scale.

1. **Performance**:

* AOSP is optimized for performance, with components engineered to deliver responsive user experiences and efficient resource utilization.
* The Android Runtime (ART) employs ahead-of-time (AOT) compilation to translate bytecode into native machine code, enhancing application startup times and runtime performance.
* AOSP's lightweight architecture and streamlined system services contribute to its overall performance, ensuring smooth operation even on resource-constrained devices.

**3.** **Technical Architecture:**

* AOSP comprises a layered architecture consisting of the Linux kernel, system libraries, application framework, and runtime environment.
* The Linux kernel provides hardware abstraction and device drivers, enabling seamless interaction between the hardware and higher-level software layers.
* The application framework offers a rich set of APIs and libraries for building Android applications, facilitating tasks such as user interface design, data storage, and network communication.
* AOSP's modular design allows for customization and extension, enabling device manufacturers and developers to tailor the platform to meet their specific requirements and preferences.

**4.** **Community and Ecosystem**

* AOSP benefits from a vibrant community of developers, device manufacturers, and enthusiasts who contribute code, provide feedback, and collaborate on the platform's development.
* Its ecosystem includes a wide range of compatible hardware devices, third-party applications, and developer tools, fostering innovation and interoperability within the Android ecosystem.
* AOSP's open-source nature encourages transparency, collaboration, and knowledge sharing, driving continuous improvement and evolution of the platform to meet the needs of users and stakeholders.

**Project Comparison**

**AOSP vs. Proprietary Mobile Operating Systems**

* **Openness and Customization:** AOSP is open-source, allowing device manufacturers and developers to customize and modify the platform according to their needs. In contrast, proprietary mobile operating systems like iOS and Windows have limited customization options and are tightly controlled by their respective companies (Apple and Microsoft).
* **Ecosystem and Compatibility:** AOSP fosters a diverse ecosystem of devices, apps, and services from various manufacturers and developers. Proprietary mobile operating systems often have a closed ecosystem, with exclusive integration of hardware, software, and services, limiting interoperability with third-party products.
* **Security and Privacy:** AOSP's open-source nature enables transparency and community scrutiny, potentially leading to faster identification and resolution of security vulnerabilities. Proprietary mobile operating systems typically have proprietary security measures and closed-source components, raising concerns about transparency and data privacy.

**AOSP vs. Custom Android ROMs (e.g., LineageOS, OxygenOS):**

* **Flexibility and Control:** AOSP provides the foundation for custom Android ROMs, offering users greater flexibility and control over their device's software experience. Custom ROMs like LineageOS and OxygenOS build upon AOSP, introducing additional features, optimizations, and customization options tailored to user preferences.
* **Community Support:** AOSP benefits from a large and active community of developers and enthusiasts who contribute to its development and support custom ROM projects. Custom ROM communities provide ongoing updates, bug fixes, and new features, extending the lifespan and usability of older devices beyond what official manufacturer support offers.

**AOSP vs. Forked Android Platforms (e.g., Amazon Fire OS, Huawei EMUI) :**

* **Independence and Differentiation :** Forked Android platforms like Amazon Fire OS and Huawei EMUI diverge from AOSP to create unique software experiences tailored to their respective hardware ecosystems. While based on AOSP, these platforms often feature custom user interfaces, app stores, and service integrations designed to differentiate themselves in the market.
* **Compatibility and App Ecosystem** : Forked Android platforms may face challenges with compatibility and app availability compared to mainstream Android devices. Differences in APIs, services, and app stores can lead to limitations in accessing certain apps or services, potentially affecting user experience and device functionality.

**References**

Repository : https://source.android.com

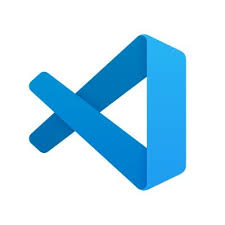
Official Website: https://android.googlesource.com/

Further Info: <https://cs.android.com/>

## 2.1 Visual Studio Code

### Introduction

Visual Studio Code (VS Code) represents Microsoft's dedication to fostering open-source development practices and empowering developers worldwide. Since its inception, VS Code has revolutionized the source-code editing landscape, offering a blend of accessibility, flexibility, and innovation that resonates with developers across diverse industries and skill levels.



**Project Summary**

|  |  |
| --- | --- |
| Website | https://code.visualstudio.com/ |
| Organization/Foundation Name | Microsoft Corporation |
| Licence | MIT License |
| Open/Proprietary | Open Source |
| Source Path | https://github.com/microsoft/vscode |
| Brief Description | VS Code redefines the developer experience with its lightweight yet robust platform. It provides a rich feature set, including comprehensive language support, a vibrant extensions ecosystem, and a modular architecture that delivers unmatched performance and flexibility. |

### Project Details

#### Key Feature:

* **Lightweight and Fast:** VS Code is engineered for speed and efficiency, offering snappy performance without compromising on functionality. Its minimalist design and optimized resource usage contribute to a smooth editing experience.
* **Built-in Language Support:** VS Code boasts extensive language support out of the box, with features such as syntax highlighting, IntelliSense (code completion), and code navigation tools that enhance productivity and streamline development workflows.
* **Extensions Ecosystem:** The VS Code marketplace hosts a vast ecosystem of extensions, themes, and plugins contributed by the community. These extensions enhance the editor's functionality, enabling developers to tailor their editing environment to suit their preferences and workflow requirements.

## Architecture

## 

1. **Main Process (Node.js):** This can be depicted as a hexagon in the centre of the diagram. Within the hexagon, list its key responsibilities:
   * Application life cycle management
   * Interaction with the operating system
   * Communication coordination between processes
2. **Renderer Process (Electron):** Draw a rectangle positioned below the Main Process hexagon. Inside the rectangle, specify its role:
   * Renders the user interface with web technologies (HTML, CSS, JavaScript)
   * Utilizes Chrome web engine for efficient rendering.
3. **Extension Host:** Create another rectangle positioned to the right of the Main Process hexagon. Label it "Extension Host" and include these details:
   * Separate process for executing extensions.
   * Isolates extensions from core VS Code functionality (improves stability and security)
4. **Node.js Modules:** Depict a cloud shape positioned above the Main Process hexagon. Fill the cloud with the text "Node.js Modules" and include some examples like:
   * File I/O
   * Networking
   * Debugging

**Current Usage**

Here's a list of notable organizations known to use Visual Studio Code (VS Code):

**1. Microsoft**: Microsoft, the organization behind the development of Visual Studio Code, utilizes it extensively for various internal and external development projects. VS Code serves as a versatile and feature-rich source-code editor within Microsoft's ecosystem.

**2. GitHub**: GitHub, a subsidiary of Microsoft and the world's largest code hosting platform, leverages Visual Studio Code to enhance its development workflow. Many GitHub repositories include configuration files tailored for VS Code, and GitHub actively contributes to the VS Code project.

**3. Google**: Google, a leading technology company, incorporates Visual Studio Code into its development toolchain for building software applications, services, and platforms. Google developers utilize VS Code's extensibility and cross-platform capabilities for coding tasks

**4. Netflix**: Netflix, a leading streaming entertainment service, incorporates Visual Studio Code into its development toolkit for building and maintaining its streaming platform. Netflix engineers leverage VS Code's features for writing and debugging code, collaborating on projects, and managing repositories.

**5. Adobe**: Adobe, a multinational software company, utilizes Visual Studio Code for various development projects, including web development, mobile app development, and creative tools development. Adobe developers benefit from VS Code's lightweight design and extensive ecosystem of extensions.

## Technical Details

**1. Scalability:** Visual Studio Code is designed to scale with the needs of developers and organizations of all sizes. Its architecture allows for horizontal scaling by leveraging distributed version control systems (e.g., Git) and cloud-based storage solutions. The lightweight nature of VS Code's core editor ensures that it can efficiently handle large codebases and projects without compromising performance. Developers can seamlessly work on projects of varying complexity and scale. Additionally, VS Code's extension ecosystem provides scalability in terms of functionality, allowing developers to enhance and customize their editing environment based on their specific requirements.

**2. Performance:** Visual Studio Code prioritizes performance to deliver a responsive and smooth editing experience. The editor's core functionalities, such as code editing, syntax highlighting, and IntelliSense, are optimized for speed and efficiency. VS Code's architecture utilizes a combination of native and web-based technologies to achieve optimal performance across different platforms. It leverages Electron for cross-platform compatibility while ensuring minimal resource usage and fast startup times. Furthermore, VS Code implements asynchronous processing techniques to handle concurrent operations efficiently. This ensures that tasks such as file operations, language services, and extension execution do not block the user interface, maintaining responsiveness during heavy workloads.

**3. Caching and Optimization:** Visual Studio Code incorporates caching mechanisms to improve performance and optimize resource utilization. This includes caching frequently accessed files, language services, and workspace state to minimize disk I/O and improve responsiveness.

The editor's built-in task management system allows developers to configure caching policies for tasks and build processes, reducing redundant computations and speeding up development workflows.

Additionally, VS Code supports various optimization techniques for frontend assets, such as bundling, minification, and code splitting. These optimizations help reduce the size of distributed extensions and improve loading times for both the editor and installed extensions.

**Project Comparison**

Here's a comparison between Backstage and GitLab:

**1. Scope and Focus:**

* **Visual Studio Code (VS Code):** VS Code is a lightweight source-code editor developed by Microsoft, focusing on providing a streamlined editing experience for developers across various programming languages and platforms.
* **GitLab:** GitLab is a comprehensive DevOps platform that encompasses source code management, CI/CD pipelines, issue tracking, project management, container registry, and more. It offers an end-to-end solution for software development and collaboration.

**2. Features:**

* **Visual Studio Code (VS Code):** VS Code offers features such as syntax highlighting, IntelliSense (code completion), debugging, version control integration, extension support, and customizable themes. It focuses on core editing capabilities and extensibility.
* GitLab: GitLab provides a wide range of features, including version control, issue tracking, CI/CD pipelines, code review, project management, and container registry.

**3.** **Integration**:

**Visual Studio Code (VS Code):** VS Code integrates with various version control systems (e.g., Git, GitHub, GitLab), CI/CD pipelines, debugging tools, and extensions from its marketplace. It seamlessly integrates with other development tools and services to enhance productivity.

**GitLab:** GitLab offers built-in integration with CI/CD pipelines, issue tracking, project management, Kubernetes, monitoring solutions, and more. It provides a unified platform for developers to collaborate and automate workflows.

**4. Customization**:

* Backstage: Backstage is highly customizable, allowing organisations to extend its functionality through plugins and customizations to meet specific requirements.
* GitLab: GitLab offers extensive customization options, allowing organisations to tailor their workflows, pipelines, and project settings to suit their needs.

**5. Community and Support:**

* **Visual Studio Code (VS Code):** VS Code is highly customizable, allowing developers to personalize their editing environment with themes, keybindings, and extensions from the marketplace. Users can customize their workflows and enhance functionality based on their preferences.
* **GitLab:** GitLab offers extensive customization options for workflows, pipelines, project settings, and user permissions. Organizations can tailor GitLab to meet their specific requirements and integrate it with third-party tools and services

### Reference:

**VS Code documentation:** [https://code.visualstudio.com/docs](https://code.visualstudio.com/docs" \t "_blank)

**VS Code release notes:** [https://code.visualstudio.com/updates](https://code.visualstudio.com/updates" \t "_blank)

**VS Code GitHub repository:** [https://github.com/microsoft/vscode](https://github.com/microsoft/vscode" \t "_blank)

**3. How to join and contribute**

**Introduction:**

Embarking on a journey to contribute to open-source projects on GitHub is an enriching experience that not only allows individuals to hone their skills but also fosters collaboration within a global community of developers. GitHub, with its vast repository of projects spanning various domains and technologies, offers an ideal platform for individuals to contribute meaningfully to projects they are passionate about. Whether you're a seasoned developer looking to give back to the community or a newcomer eager to learn and grow, GitHub provides ample opportunities to make a difference. By joining in contributing to GitHub projects, you become part of a vibrant ecosystem where innovation thrives, knowledge is shared, and collective efforts propel projects forward.

Joining and contributing to various communities or projects can be a fulfilling way to make a positive impact. Here are some general steps you can follow:

* Identify your interests and passions
* Research organizations or communities
* Understand their mission and values
* Explore different roles and opportunities
* Reach out and express interest
* Volunteer your time and skills
* Collaborate and participate actively
* Learn and grow
* Seek feedback and take on challenges
* Maintain commitment and sustainability

**Entry Channels:**

* Online Platforms
* Application Processes
* Networking and Referrals
* Events and Workshops
* Volunteer Programs
* Internship and Apprenticeship Programs
* Professional Associations and Societies
* Academic Institutions .

**How to contribute:**

* Volunteer your time
* Share your expertise
* Join committees or working groups
* Contribute to research or content creation
* Support with fundraising efforts
* Offer technical or digital assistance

**Attend community meeting and experience sharing:**

* Identify relevant community meetings
* Check event listings and announcements
* RSVP or register
* Prepare beforehand
* Attend the meeting
* Engage actively
* Take notes and network

**Tools and Setup in open source:**

* Version Control Systems (VCS)
* Issue Tracking Systems
* Collaboration and Communication Platforms
* Continuous Integration/Continuous Deployment (CI/CD) Tools
* Code Review Tools
* Documentation Tools
* Development Environments
* Testing and Code Quality Tools
* Build and Dependency Management Tools:
* Virtualization and Containerization

**Contribution and understand various aspects of contribution:**

* Code Contributions
* Documentation
* Bug Reporting and Issue Triage
* Testing and Quality Assurance
* User Support and Community Engagement
* Design and User Experience (UX)
* Translation and Localization
* Community Leadership and Project Management

**Quality in open source:**

* Clear Project Vision and Goals
* Community Engagement
* Testing and Quality Assurance
* Documentation
* Bug Tracking and Issue Management
* Continuous Integration/Continuous Deployment (CI/CD)
* Security and Vulnerability Management
* User Feedback and Usability Testing
* Release Management
* Continuous Improvement

**Testing open source:**

* Test Strategy and Planning
* Unit Testing
* Integration Testing
* Functional Testing
* Performance Testing
* Security Testing
* Continuous Testing
* Bug Tracking and Issue Management
* Collaboration and Code Review
* User Feedback and Beta Testing
* Documentation

**Documentation in open source:**

* README File
* Installation and Setup Guides
* Usage Examples and Tutorials
* API and Developer Documentation
* Contribution Guidelines
* Versioning and Changelog
* User Support and Community Engagement
* Keeping Documentation Up to Date
* Documentation Formats and Tools

**Best Practices Sharing open source**

* Document and Share
* Use a Version Control System
* Promote Consistency
* Foster Discussion and Feedback
* Highlight Examples and Use Cases
* Encourage Continuous Improvement
* Recognize and Attribute Contributors
* Integrate Best Practices in Onboarding
* Provide Tooling and Automation
* Lead by Example

**Conclusion:**

In conclusion, contributing to GitHub projects is not just about writing code or fixing bugs; it's about being part of something bigger than oneself. It's about collaborating with like-minded individuals, sharing knowledge, and making a positive impact on the world of software development. Whether you're adding new features, improving documentation, or providing support to users, every contribution matters and contributes to the collective success of the open-source community. So, if you're ready to dive in and be part of this exciting journey, GitHub welcomes you with open arms. Join the millions of developers around the globe who are shaping the future of technology, one contribution at a time.

### 4.1 Meeting Report

**Topic 1 :**  SODA Global Community Meeting

**Date** : 02/02/2024

**Time** : 12pm-1pm

#### Discussion

The Soda Global Community Meeting brought together students from diverse backgrounds and cultures in an enthusiastic gathering aimed at fostering collaboration, exchange, and learning. The virtual event served as a vibrant platform for participants to connect and engage in meaningful ways. Key highlights of the meeting included:

* **Collaboration Opportunities:** Students were presented with numerous chances to partner on interdisciplinary projects focused on sustainability, technology, and innovation. Dynamic discussions sparked ideas for impactful partnerships.
* **Cultural Exchange:** Attendees celebrated diversity by sharing insights into their traditions, customs, and perspectives. This promoted mutual understanding and enriched the collective experience.
* **Skill Development Workshops:** Interactive sessions covered leadership, communication, problem-solving, and more. These empowered students with practical tools to effectively navigate academic and professional challenges.
* **Guest Speaker Sessions:** Renowned experts delivered inspiring talks on global trends, social entrepreneurship, and other topics. Their insights resonated with attendees, sparking thought-provoking conversations and passion for change.
* **Networking Opportunities:** The meeting enabled valuable connections between students and mentors from around the world. These relationships can lead to future collaborations and career advancement.

Overall, the event successfully fostered a sense of community, collaboration, and empowerment. Looking ahead, this spirit of camaraderie.

### 4.2 Tech Talk report

**Topic 2** : Embracing Open Source : Your Path to Contribute

**Date** : 29/02/2024

**Time** : 1:40 pm – 3:40 pm

#### Discussion :

A Zoom meeting hosted by Ashit Kumar, Software Development Manager at Amazon Alexa, on 29/02/2024 focused on the Sofa Foundation and its open-source contribution opportunities. With 17 years in the industry, including roles at Huawei and Veritas Technologies, Ashit has deep knowledge that he shared.

Ashit highlighted the importance of open-source projects, drawing on his experience as a Sofa Foundation maintainer. He emphasized the value of collaboration in open-source communities and its impact on technological progress.

Much of the meeting illuminated the Sofa Foundation, including its goals of promoting open collaboration and innovation for cloud-native software. Ashit described the foundation's commitment to an inclusive environment for diverse contributors.

Additionally, Ashit shared insights from his work as a visiting faculty member at Pune University's Computer Science Department, where he educates students on open-source projects like the Sofa Foundation. He stressed the role of education in fostering a culture of contribution and encouraged engaging educational institutions to promote open-source awareness.

The meeting then focused on practical guidance for contributing to Sofa Foundation projects. Ashit provided advice on identifying suitable projects, understanding documentation, and navigating the contribution process effectively. He highlighted the importance of communication within the community and utilizing resources like forums and mailing lists.

In summary, the meeting provided a comprehensive overview of Sofa Foundation contribution opportunities, with Ashit Kumar's expertise enabling meaningful open-source community engagement.