**MAD PROJECT CODE COMPLETE ( RECIPE GENERATOR USING GEMINI AI API)**

**Roll no: 220701311  
Class and Sec: CSE – D**

**Name: Varun Kumar V**

#### PROGRAM CODE AndroidManifest.xml

|  |
| --- |
| **<?xml version="1.0" encoding="utf-8"?>** |
| **<manifest xmlns:android="http://schemas.android.com/apk/res/android"** |
| **xmlns:tools="http://schemas.android.com/tools">** |
|  |
| **<application** |
| **android:allowBackup="true"** |
| **android:dataExtractionRules="@xml/data\_extraction\_rules"** |
| **android:fullBackupContent="@xml/backup\_rules"** |
| **android:icon="@mipmap/ic\_launcher"** |
| **android:label="@string/app\_name"** |
| **android:roundIcon="@mipmap/ic\_launcher\_round"** |
| **android:supportsRtl="true"** |
| **android:theme="@style/Theme.GitHelp"** |
| **tools:targetApi="31">** |
| **<activity** |
| **android:name=".SplashScreen"** |
| **android:exported="true">** |
| **<intent-filter>** |
| **<action android:name="android.intent.action.MAIN" />** |
|  |
| **<category android:name="android.intent.category.LAUNCHER" />** |
| **</intent-filter>** |
| **</activity>** |
| **<activity** |
| **android:name=".MainActivity"** |
| **android:exported="true"** |
| **android:windowSoftInputMode="stateHidden" />** |
| **</application>** |
|  |
| **</manifest>** |

#### activity\_main.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> |
| <layout xmlns:android="http://schemas.android.com/apk/res/android" |
| xmlns:app="http://schemas.android.com/apk/res-auto" |
| xmlns:tools="http://schemas.android.com/tools"> |
|  |
| <ScrollView |
| android:layout\_width="match\_parent" |
| android:layout\_height="match\_parent"> |
|  |
| <androidx.constraintlayout.widget.ConstraintLayout |
| android:layout\_width="match\_parent" |
| android:layout\_height="wrap\_content" android:padding="16dp" |
| tools:context=".MainActivity"> |
|  |
| <TextView |
| android:id="@+id/text\_git\_command" |
| android:layout\_width="wrap\_content" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="44dp" |
| android:fontFamily="@font/calibri" |
| android:textSize="24sp" |
| app:layout\_constraintEnd\_toEndOf="parent" |
| app:layout\_constraintStart\_toStartOf="parent" |
| app:layout\_constraintTop\_toTopOf="parent" |

|  |
| --- |
| tools:text="Git Command Explorer" /> |
|  |
| <TextView |
| android:id="@+id/text\_find\_right\_command" |
| android:layout\_width="wrap\_content" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="16dp" |
| android:fontFamily="@font/calibri" android:gravity="center" |
| android:text="Find the right commands you need without digging |
| through the web" |
| android:textColor="#6A6A6A" |
| android:textSize="16sp" |
| app:layout\_constraintEnd\_toEndOf="parent" |
| app:layout\_constraintStart\_toStartOf="parent" |
| app:layout\_constraintTop\_toBottomOf="@+id/text\_git\_command" /> |
|  |
| <TextView |
| android:id="@+id/text\_i\_want\_to" |
| android:layout\_width="wrap\_content" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="44dp" |
| android:fontFamily="@font/calibri" |
| android:text="I want to:" |
| android:textColor="@color/colorAccent" |
| android:textSize="18sp" |
| android:textStyle="bold" |
| app:layout\_constraintStart\_toStartOf="parent" |
|  |
| app:layout\_constraintTop\_toBottomOf="@+id/text\_find\_right\_command" /> |
|  |

|  |
| --- |
| <androidx.cardview.widget.CardView |
| android:id="@+id/card\_view\_first\_field" |
| android:layout\_width="0dp" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="16dp" |
| app:cardCornerRadius="4dp" |
| app:cardUseCompatPadding="true" |
| app:layout\_constraintEnd\_toEndOf="parent" app:layout\_constraintStart\_toStartOf="parent" |
| app:layout\_constraintTop\_toBottomOf="@+id/text\_i\_want\_to"> |
|  |
| <AutoCompleteTextView |
| android:id="@+id/input\_first\_field" |
| android:layout\_width="match\_parent" |
| android:layout\_height="wrap\_content" |
| android:background="@null" |
| android:completionThreshold="1" |
| android:drawableEnd="@drawable/ic\_keyboard\_arrow\_down" |
| android:drawableTint="#ACACAC" |
| android:fontFamily="@font/calibri" |
| android:hint="..." |
| android:inputType="textNoSuggestions" |
| android:padding="12dp" |
| android:textColor="@color/colorPrimary" |
| android:textSize="14sp" |
| android:textStyle="bold" |
| tools:text="add" /> |
|  |
| </androidx.cardview.widget.CardView> |
|  |

|  |
| --- |
| <androidx.cardview.widget.CardView |
| android:id="@+id/card\_view\_second\_field" |
| android:layout\_width="0dp" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="4dp" |
| android:visibility="gone" |
| app:cardCornerRadius="4dp" |
| app:cardUseCompatPadding="true" app:layout\_constraintEnd\_toEndOf="parent" |
| app:layout\_constraintStart\_toStartOf="parent" |
|  |
| app:layout\_constraintTop\_toBottomOf="@+id/card\_view\_first\_field"> |
|  |
| <AutoCompleteTextView |
| android:id="@+id/input\_second\_field" |
| android:layout\_width="match\_parent" |
| android:layout\_height="wrap\_content" |
| android:background="@null" |
| android:completionThreshold="1" |
| android:drawableEnd="@drawable/ic\_keyboard\_arrow\_down" |
| android:drawableTint="#ACACAC" |
| android:fontFamily="@font/calibri" |
| android:hint="..." |
| android:inputType="textNoSuggestions" |
| android:padding="12dp" |
| android:textColor="@color/colorPrimary" |
| android:textSize="14sp" |
| android:textStyle="bold" |
| tools:text="new changes" /> |
|  |
| </androidx.cardview.widget.CardView> |

|  |
| --- |
|  |
| <TextView |
| android:id="@+id/text\_usage" |
| android:layout\_width="wrap\_content" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="24dp" |
| android:fontFamily="@font/calibri" |
| android:text="Usage" android:textColor="@color/colorPrimary" |
| android:textSize="18sp" |
| android:textStyle="bold" |
| app:layout\_constraintStart\_toStartOf="parent" |
|  |
| app:layout\_constraintTop\_toBottomOf="@+id/card\_view\_second\_field" /> |
|  |
| <androidx.cardview.widget.CardView |
| android:id="@+id/card\_view\_usage" |
| android:layout\_width="0dp" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="8dp" |
| app:cardBackgroundColor="#20262C" |
| app:cardCornerRadius="8dp" |
| app:cardUseCompatPadding="true" |
| app:layout\_constraintEnd\_toEndOf="parent" |
| app:layout\_constraintStart\_toStartOf="parent" |
| app:layout\_constraintTop\_toBottomOf="@+id/text\_usage"> |
|  |
| <LinearLayout |
| android:layout\_width="match\_parent" |
| android:layout\_height="wrap\_content" |
| android:orientation="horizontal"> |

|  |
| --- |
|  |
| <View |
| android:layout\_width="0dp" |
| android:layout\_height="match\_parent" |
| android:layout\_weight="2" |
| android:background="@color/colorAccent" /> |
|  |
| <TextView android:id="@+id/text\_display\_git\_command" |
| android:layout\_width="0dp" |
| android:layout\_height="wrap\_content" |
| android:layout\_weight="98" |
| android:fontFamily="@font/calibri" |
| android:lineSpacingExtra="8dp" |
| android:padding="16dp" |
| android:textColor="@color/white" |
| android:textSize="14sp" |
| android:textStyle="bold" |
| tools:text="git add file.ext" /> |
|  |
| </LinearLayout> |
|  |
|  |
| </androidx.cardview.widget.CardView> |
|  |
|  |
| <TextView |
| android:id="@+id/text\_note" |
| android:layout\_width="wrap\_content" |
| android:layout\_height="wrap\_content" |

|  |
| --- |
| android:layout\_marginTop="16dp" |
| android:fontFamily="@font/calibri" |
| android:text="Note" |
| android:textColor="@color/colorPrimary" |
| android:textSize="18sp" |
| android:textStyle="bold" |
| android:visibility="gone" |
| app:layout\_constraintStart\_toStartOf="parent" app:layout\_constraintTop\_toBottomOf="@+id/card\_view\_usage" /> |
|  |
| <androidx.cardview.widget.CardView |
| android:id="@+id/card\_view\_note" |
| android:layout\_width="0dp" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="8dp" |
| android:visibility="gone" |
| app:cardBackgroundColor="#20262C" |
| app:cardCornerRadius="8dp" |
| app:cardUseCompatPadding="true" |
| app:layout\_constraintEnd\_toEndOf="parent" |
| app:layout\_constraintStart\_toStartOf="parent" |
| app:layout\_constraintTop\_toBottomOf="@+id/text\_note"> |
|  |
| <LinearLayout |
| android:layout\_width="match\_parent" |
| android:layout\_height="wrap\_content" |
| android:orientation="horizontal"> |
|  |
| <View |
| android:layout\_width="0dp" |
| android:layout\_height="match\_parent" |
| android:layout\_weight="2" |
| android:background="@color/colorAccent" /> |
|  |
| <TextView |
| android:id="@+id/text\_display\_note" |
| android:layout\_width="0dp" |
| android:layout\_height="wrap\_content" android:layout\_weight="98" |
| android:fontFamily="@font/calibri" |
| android:lineSpacingExtra="8dp" |
| android:padding="16dp" |
| android:textColor="@color/white" |
| android:textSize="14sp" |
| android:textStyle="bold" |
| tools:text="To add all the files in the current |
| directory, use 'git add' . To add a directory use 'git add directory'" /> |
|  |
| </LinearLayout> |
|  |
|  |
| </androidx.cardview.widget.CardView> |
|  |
| </androidx.constraintlayout.widget.ConstraintLayout> |
|  |
| </ScrollView> |
| </layout> |

#### activity\_splash\_screen.xml

|  |
| --- |
| **<?xml version="1.0" encoding="utf-8"?>** |
| **<layout xmlns:android="http://schemas.android.com/apk/res/android"** |
| **xmlns:app="http://schemas.android.com/apk/res-auto"** |
| **xmlns:tools="http://schemas.android.com/tools">** |
|  |
|  |
| **<androidx.constraintlayout.widget.ConstraintLayout** |
| **android:layout\_width="match\_parent" android:layout\_height="match\_parent"** |
| **android:padding="16dp"** |
| **tools:context=".SplashScreen">** |
|  |
| **<TextView** |
| **android:id="@+id/text\_git\_command"** |
| **android:layout\_width="wrap\_content"** |
| **android:layout\_height="wrap\_content"** |
| **android:fontFamily="@font/calibri"** |
| **android:textSize="24sp"** |
| **app:layout\_constraintBottom\_toBottomOf="parent"** |
| **app:layout\_constraintEnd\_toEndOf="parent"** |
| **app:layout\_constraintStart\_toStartOf="parent"** |
| **app:layout\_constraintTop\_toTopOf="parent"** |
| **tools:text="Git Command Explorer" />** |
|  |
| **<TextView** |
| **android:id="@+id/text\_find\_right\_command"** |
| **android:layout\_width="wrap\_content"** |
| **android:layout\_height="wrap\_content"** |
| **android:layout\_marginTop="16dp"** |
| **android:fontFamily="@font/calibri"** |
| **android:gravity="center"** |
| **android:text="Find the right commands you need without digging** |
| **through the web"** |
| **android:textColor="#6A6A6A"** |
| **android:textSize="16sp"** |
| **app:layout\_constraintEnd\_toEndOf="parent"** |
| **app:layout\_constraintStart\_toStartOf="parent"** |
| **app:layout\_constraintTop\_toBottomOf="@+id/text\_git\_command" />** |
| **</androidx.constraintlayout.widget.ConstraintLayout>** |
| **</layout>** |

#### git\_command\_explorer.json

|  |
| --- |
| **{** |
| **"primary\_options": [** |
| **{** |
| **"value": "add",** |
| **"label": "add"** |
| **},** |
| **{** |
| **"value": "commit",** |
| **"label": "commit"** |
| **},** |
| **{** |
| **"value": "revert",**  **"label": "revert/reset"** |
| **},** |
| **{** |
| **"value": "initialize",** |
| **"label": "initialize"** |
| **},** |
| **{** |

|  |
| --- |
| **"value": "modify",** |
| **"label": "modify"** |
| **},** |
| **{** |
| **"value": "show",** |
| **"label": "show/view"** |
| **},** |
| **{**  **"value": "delete",** |
| **"label": "delete/remove"** |
| **},** |
| **{** |
| **"value": "compareCommits",** |
| **"label": "compare two commits"** |
| **},** |
| **{** |
| **"value": "configure",** |
| **"label": "configure"** |
| **},** |
| **{** |
| **"value": "clone",** |
| **"label": "clone"** |
| **},** |
| **{** |
| **"value": "ignore",** |
| **"label": "ignore"** |
| **},** |
| **{** |
| **"value": "rename",** |
| **"label": "rename"** |

|  |
| --- |
| **},** |
| **{** |
| **"value": "merge",** |
| **"label": "merge"** |
| **},** |
| **{** |
| **"value": "squash",** |
| **"label": "squash"**  **},** |
| **{** |
| **"value": "stash",** |
| **"label": "stash"** |
| **},** |
| **{** |
| **"value": "debug",** |
| **"label": "debug"** |
| **},** |
| **{** |
| **"value": "recover",** |
| **"label": "recover"** |
| **},** |
| **{** |
| **"value": "synchronize",** |
| **"label": "synchronize"** |
| **},** |
| **{** |
| **"value": "rebase",** |
| **"label": "rebase"** |
| **}** |
| **],** |

|  |
| --- |
| **"secondary\_options": {** |
| **"commit": [** |
| **{** |
| **"value": "local-changes",** |
| **"label": "commit all local changes in tracked files",** |
| **"usage": "git commit -a"** |
| **},** |
| **{**  **"value": "staged-changes",** |
| **"label": "commit all staged changes",** |
| **"usage": "git commit -m <message>",** |
| **"nb": "Replace <message> with your commit message."** |
| **}** |
| **],** |
| **"configure": [** |
| **{** |
| **"value": "email-name",** |
| **"label": "name and email address",** |
| **"usage": "git config --global user.name \"username\" \n\ngit config** |
| **--global user.email \"email address\"",** |
| **"nb": "Your username and email address should be the same as the one** |
| **used with your git hosting provider i.e. github, bitbucket, gitlab etc"** |
| **},** |
| **{** |
| **"value": "editor",** |
| **"label": "default editor",** |
| **"usage": "git config --global core.editor \"vim\"",** |
| **"nb": "Change default editor to vim."** |
| **},** |
| **{** |
| **"value": "diff-tool",** |

|  |
| --- |
| **"label": "external diff tool",** |
| **"usage": "git config --global diff.external \"meld\"",** |
| **"nb": "Set external diff tool to meld."** |
| **},** |
| **{** |
| **"value": "merge-tool",** |
| **"label": "default merge tool",** |
| **"usage": "git config --global merge.tool \"meld\"",**  **"nb": "Set default merge tool to meld."** |
| **},** |
| **{** |
| **"value": "color",** |
| **"label": "color",** |
| **"usage": "git config --global color.ui auto",** |
| **"nb": "Enables helpful colorization of command line output"** |
| **},** |
| **{** |
| **"value": "signingkey",** |
| **"label": "add the GPG key",** |
| **"usage": "git config --global user.signingkey <your-secret-gpg-key>",** |
| **"nb": "Git is cryptographically secure, but it’s not foolproof. If** |
| **you’re taking work from others on the internet and want to verify that** |
| **commits are actually from a trusted source, Git has a few ways to sign and** |
| **verify work using GPG."** |
| **}** |
| **],** |
| **"revert": [** |
| **{** |
| **"value": "specific-commit",** |
| **"label": "a specific commit",** |
| **"usage": "git revert <commit-hash>",** |
| **"nb": "Use git log to see the hash of each commit"** |

|  |
| --- |
| **},** |
| **{** |
| **"value": "to-last-commit",** |
| **"label": "to last commit",** |
| **"usage": "git reset --hard"** |
| **},** |
| **{** |
| **"value": "to-last-commit-from-remote",**  **"label": "to last commit on remote branch",** |
| **"usage": "git reset --hard <repo>/<branch>"** |
| **}** |
| **],** |
| **"initialize": [** |
| **{** |
| **"value": "new-repo",** |
| **"label": "a new repository",** |
| **"nb": "Make sure you are in the right directory",** |
| **"usage": "git init"** |
| **}** |
| **],** |
| **"modify": [** |
| **{** |
| **"value": "commit-message",** |
| **"label": "my last/latest commit message",** |
| **"usage": "git commit --amend"** |
| **},** |
| **{** |
| **"value": "commit",** |
| **"label": "my last commit but leave the commit message as is",** |
| **"usage": "git add . \ngit commit --amend --no-edit"** |

|  |
| --- |
| **},** |
| **{** |
| **"value": "remoteUrl",** |
| **"label": "repo's remote url",** |
| **"usage": "git remote set-url <alias> <url>",** |
| **"nb": "<alias> is your remote name e.g origin"** |
| **}** |
| **],**  **"show": [** |
| **{** |
| **"value": "repo-status",** |
| **"label": "status of project including staged, unstaged and untracked** |
| **files",** |
| **"usage": "git status"** |
| **},** |
| **{** |
| **"value": "logs",** |
| **"label": "commit logs/history"** |
| **},** |
| **{** |
| **"value": "uncommittedChanges",** |
| **"label": "uncommitted changes",** |
| **"usage": "git diff"** |
| **},** |
| **{** |
| **"value": "committedChanges",** |
| **"label": "committed/staged changes",** |
| **"usage": "git diff --staged"** |
| **},** |
| **{** |
| **"value": "remoteUrl",** |

|  |
| --- |
| **"label": "repo's remote url",** |
| **"usage": "git remote -v"** |
| **},** |
| **{** |
| **"value": "stash",** |
| **"label": "stash",** |
| **"usage": "git stash list"** |
| **},**  **{** |
| **"value": "branch",** |
| **"label": "branches",** |
| **"usage": "git branch",** |
| **"nb": "The active branch is prefixed with \*"** |
| **},** |
| **{** |
| **"value": "tags",** |
| **"label": "tags",** |
| **"usage": "git tag"** |
| **}** |
| **],** |
| **"delete": [** |
| **{** |
| **"value": "branch",** |
| **"label": "a branch",** |
| **"usage": "git branch -D <branch name>"** |
| **},** |
| **{** |
| **"value": "delete-multiple-branches",** |
| **"label": "multiple branches"** |
| **},** |

|  |
| --- |
| **{** |
| **"value": "tag",** |
| **"label": "a tag",** |
| **"usage": "git tag -d v<tag version>"** |
| **},** |
| **{** |
| **"value": "remote",** |
| **"label": "remote",**  **"usage": "git remote rm <remote>"** |
| **},** |
| **{** |
| **"value": "untracked-files",** |
| **"label": "untracked files",** |
| **"usage": "git clean -<flag>",** |
| **"nb": "replace -<flag> with:\n -i for interactive command\n -n to** |
| **preview what will be removed\n -f to remove forcefully\n -d to remove** |
| **directories\n -X to remove ignored files\n -x to remove ignored and** |
| **non-ignored files"** |
| **},** |
| **{** |
| **"value": "files-from-index",** |
| **"label": "files from index",** |
| **"usage": "git rm --cached <file or dir>",** |
| **"nb": "Use this option to unstage and remove paths only from the** |
| **index. Working tree files, whether modified or not, will be left alone."** |
| **},** |
| **{** |
| **"value": "local-branches-not-on-remote",** |
| **"label": "local branches that don't exist at remote",** |
| **"usage": "git remote prune <remote-name>",** |
| **"nb": "Use the --dry-run option to report what branches will be** |
| **pruned, but do not actually prune them"** |
| **},** |

|  |
| --- |
| **{** |
| **"value": "files-from-old-commit",** |
| **"label": "files from old commits",** |
| **"usage": "git filter-branch --index-filter \n'git rm --cached** |
| **--ignore-unmatch path/to/mylarge\_file' \n--tag-name-filter cat --** |
| **--all\n\nfilter-branch keeps backups too, so the size of the repo won't** |
| **decrease immediately unless you expire the reflogs and garbage collect:\n\nrm** |
| **-Rf .git/refs/original # careful\ngit gc --aggressive --prune=now #** |
| **danger",** |
| **"nb": "Like the rebasing option described before, filter-branch is** |
| **rewriting operation. If you have published history, you'll have to --force** |
| **push the new refs."** |
| **}** |
| **],** |
| **"compareCommits": [** |
| **{** |
| **"value": "terminal",** |
| **"label": "and output result in the terminal",** |
| **"usage": "git diff <sha1> <sha2> | less",** |
| **"nb": "sha1 and sha2 are the sha hash of the commits you want to** |
| **compare."** |
| **},** |
| **{** |
| **"value": "file",** |
| **"label": "and output result to a file",** |
| **"usage": "git diff <sha1> <sha2> > diff.txt",** |
| **"nb": "sha1 and sha2 are the sha of the commits you want to compare.** |
| **\n\ndiff.txt is the file you want to store the contents of the diff"** |
| **}** |
| **],** |
| **"clone": [** |
| **{** |
| **"value": "clone-repo-into-a-new-dir",** |
| **"label": "existing repo into a new directory",** |
| **"usage": "git clone <repo-url> <directory>",** |

|  |
| --- |
| **"nb": "The repo is cloned into the specified directory\n\nReplace** |
| **\"directory\" with the directory you want"** |
| **},** |
| **{** |
| **"value": "clone-repo-into-a-current-dir",** |
| **"label": "existing repo into the current directory",** |
| **"usage": "git clone <repo-url> .",** |
| **"nb": "The repo is cloned into the current directory\n\nThe current** |
| **directory is represented with a \".\" (period)"** |
| **},** |
| **{** |
| **"value": "clone-repo-with-submodule-into-a-current-dir",** |
| **"label": "existing repo along with submodules into the current** |
| **directory",** |
| **"usage": "git clone --recurse-submodules <repo-url> .",** |
| **"nb": "If git version is under 2.13, use --recursive option instead."** |
| **},** |
| **{** |
| **"value": "clone-submodule-after",** |
| **"label": "submodules after cloning existing repo",** |
| **"usage": "git submodule update --init --recursive"** |
| **}** |
| **],** |
| **"ignore": [** |
| **{** |
| **"value": "ignore-files-in-a-dir",** |
| **"label": "all files in a directory",** |
| **"usage": "<dir name>/\*",** |
| **"nb": "This must be added to .gitignore file\n\nReplace \"dir name\"** |
| **with name of directory whose files you want git to ignore"** |
| **},** |
| **{** |

|  |
| --- |
| **"value": "ignore-all-files-of-a-specific-type",** |
| **"label": "all files of a specific type",** |
| **"usage": "\*.<filename extension>",** |
| **"nb": "This must be added to .gitignore file\n\nReplace \"filename** |
| **extension\" with the extension of the files you want git to ignore\n\nFor** |
| **example \*.py tells git to ignore all python files in the repository"** |
| **}** |
| **],** |
| **"help": [** |
| **{** |
| **"value": "command-help",** |
| **"label": "about a command",** |
| **"usage": "append --help to the command",** |
| **"nb": "e.g. git merge --help\n\nType q to quite terminal"** |
| **}** |
| **],** |
| **"add": [** |
| **{** |
| **"value": "new-changes",** |
| **"label": "new changes",** |
| **"usage": "git add <file.ext>",** |
| **"nb": "To add all the files in the current directory, use \"git add**  **.\"\n\nTo add a directory use \"git add <directory>\""** |
| **},** |
| **{** |
| **"value": "add-new-branch",** |
| **"label": "a new branch",** |
| **"usage": "git branch <branch-name>"** |
| **},** |
| **{** |
| **"value": "add-repo",** |
| **"label": "new remote repo",** |

|  |
| --- |
| **"usage": "git remote add <shortname> <url>"** |
| **},** |
| **{** |
| **"value": "add-alias",** |
| **"label": "alias",** |
| **"usage": "git config --global alias.<alias> <command>",** |
| **"nb": "e.g. git config --global alias.st status. Typing git st in the** |
| **terminal now does the same thing as git status"** |
| **},** |
| **{** |
| **"value": "add-annotated-tag",** |
| **"label": "annotated tag",** |
| **"usage": "git tag -a v1.4 -m \"my version 1.4\"\n\ngit push --tags"** |
| **},** |
| **{** |
| **"value": "add-annotated-tag-for-old-commit",** |
| **"label": "annotated tag for old commit",** |
| **"usage": "git tag -a v1.2 -m 'version 1.2' <commit-hash>\n\ngit push** |
| **--tags"** |
| **}** |
| **],** |
| **"push": [**  **{** |
| **"value": "new-remote-branch",** |
| **"label": "non-existent remote branch",** |
| **"usage": "git push -u origin <branchname>"** |
| **}** |
| **],** |
| **"rename": [** |
| **{** |
| **"value": "branch",** |

|  |
| --- |
| **"label": "branch"** |
| **},** |
| **{** |
| **"value": "file",** |
| **"label": "file",** |
| **"usage": "git mv file\_from file\_to"** |
| **},** |
| **{**  **"value": "remoteUrl",** |
| **"label": "remote",** |
| **"usage": "git remote rename <oldname> <newname>"** |
| **}** |
| **],** |
| **"merge": [** |
| **{** |
| **"value": "branch",** |
| **"label": "another branch to current branch",** |
| **"usage": "git merge <branch-name>"** |
| **},** |
| **{** |
| **"value": "single-file",** |
| **"label": "merge a single file from one branch to another.",** |
| **"usage": "git checkout <branch name> <path to file> --patch"** |
| **}** |
| **],** |
| **"squash": [** |
| **{** |
| **"value": "pr",** |
| **"label": "commits in pull request into single commit",** |
| **"usage": "git rebase -i <branch name>",** |

|  |
| --- |
| **"nb": "Make sure that latest commits are fetched from upstream.\n\nFor** |
| **example (assuming you have a remote named upstream):\n\ngit fetch** |
| **upstream\ngit rebase -i upstream/master\n\nChange \"pick\" to \"squash\" for** |
| **the commits you wish to squash and save.\n\ngit push origin <topic branch>** |
| **--force-with-lease"** |
| **},** |
| **{** |
| **"value": "commits",** |
| **"label": "last n number of commit into one",** |
| **"usage": "git reset --soft HEAD~N\ngit add .\ngit commit -m** |
| **<message>",** |
| **"nb": "Replace N with the number of commits you want to squash and** |
| **<message> with your commit message. You can use the command \"git log\" to** |
| **view your commit history"** |
| **}** |
| **],** |
| **"debug": [** |
| **{** |
| **"value": "bisect",** |
| **"label": "binary search",** |
| **"usage": "git bisect start\ngit bisect bad # Current** |
| **version is bad\ngit bisect good v2.13 # v6.12 is known to be good",** |
| **"nb": "Once you have specified at least one bad and one good commit,** |
| **git bisect selects a commit in the middle of that range of history, checks it** |
| **out, and outputs something similar to the following:\nBisecting: 675** |
| **revisions left to test after this (roughly 10 steps)\nYou should now compile** |
| **the checked-out version and test it. If that version works correctly,** |
| **type\n\ngit bisect good\n\nIf that version is broken, type\n\ngit bisect** |
| **bad\n\nThen git bisect will respond with something like\n\nBisecting: 337** |
| **revisions left to test after this (roughly 9 steps)\n\nKeep repeating the** |
| **process: compile the tree, test it, and depending on whether it is good or** |
| **bad run git bisect good or git bisect bad to ask for the next commit that** |
| **needs testing.\nEventually there will be no more revisions left to inspect,** |
| **and the command will print out a description of the first bad commit. The** |
| **reference refs/bisect/bad will be left pointing at that commit.\nAfter a** |
| **bisect session, to clean up the bisection state and return to the original** |
| **HEAD, issue the following command:\n\ngit bisect reset"** |
| **},** |
| **{** |
| **"value": "blame",** |
| **"label": "who modified each lines",** |
| **"usage": "git blame -L <number-line-start>,<number-line-end>** |

|  |
| --- |
| **<filename>",** |
| **"nb": "The -L option will restrict the output to the requested line** |
| **range\n"** |
| **},** |
| **{** |
| **"value": "grep",** |
| **"label": "search in files",** |
| **"usage": "git grep -n <your\_text\_or\_expression>",** |
| **"nb": "Print lines matching a pattern.\nOption -n to display the** |
| **numbering of lines in which there are matches"** |
| **}** |
| **],** |
| **"recover": [** |
| **{** |
| **"value": "dropped-commit",** |
| **"label": "show hashes dangling commits after hard reset to previous** |
| **commit",** |
| **"usage": "git reflog",** |
| **"nb": "alternative: git log -g. For recovery use\ngit checkout -b** |
| **<recovery> <hash>"** |
| **},** |
| **{** |
| **"value": "deleted-branch",** |
| **"label": "show hashes removed branch or other git objects",** |
| **"usage": "git fsck --full",** |
| **"nb": "show hashes all dangling git objects. For recovery use\ngit** |
| **checkout -b <recovery> <hash>"** |
| **}** |
| **],** |
| **"rebase": [** |
| **{** |
| **"value": "origin-branch",** |
| **"label": "an origin branch into my working branch",** |

|  |
| --- |
| **"usage": "git pull --rebase origin <branch name>",** |
| **"nb": "Rebase an origin branch into working branch. Replace <branch** |
| **name> with the branch you are pulling"** |
| **},** |
| **{** |
| **"value": "local-branch",** |
| **"label": "a local branch into my working branch",** |
| **"usage": "git pull --rebase <branch name>",** |
| **"nb": "Rebase another local branch into working branch. Replace** |
| **<branch name> with the branch you are pulling"** |
| **},** |
| **{** |
| **"value": "skip",** |
| **"label": "and skip a commit",** |
| **"usage": "git rebase --skip",** |
| **"nb": "During rebase, git might not be able to automatically apply** |
| **commits due to conflicts. You can use this command to discard of your own** |
| **changes in the current commit and apply the changes from an incoming branch"** |
| **},** |
| **{** |
| **"value": "continue",** |
| **"label": "and continue after resolving conflicts",** |
| **"usage": "git rebase --continue",** |
| **"nb": "During rebase, git might not be able to automatically apply** |
| **commits due to conflicts. You can resolve this conflicts manually and use** |
| **this command to continue your rebase operation"** |
| **}** |
| **],** |
| **"synchronize": [** |
| **{** |
| **"value": "branch-from-fork",** |
| **"label": "a branch in a fork",** |
| **"usage": "git fetch <remote-repo> \n\ngit checkout <branch-name>** |
| **\n\ngit merge <remote-repo>/<branch-name>",** |

|  |
| --- |
| **"nb": "You need to add a remote repo for your fork first."** |
| **}** |
| **],** |
| **"stash": [** |
| **{** |
| **"value": "save-stash",** |
| **"label": "(un)tracked files",** |
| **"usage": "git stash",**  **"nb": "To stash with a customized message use git stash save** |
| **<message>\n\nTo stash untracked files git stash save -u"** |
| **},** |
| **{** |
| **"value": "list-stash",** |
| **"label": "view list of stashed changes",** |
| **"usage": "git stash list"** |
| **},** |
| **{** |
| **"value": "apply-stash",** |
| **"label": "apply"** |
| **},** |
| **{** |
| **"value": "show",** |
| **"label": "view the contents of a stash",** |
| **"usage": "git stash show -p <stash id>",** |
| **"nb": "You can leave out the stash id if you want the contents of the** |
| **latest stash"** |
| **},** |
| **{** |
| **"value": "delete-stash",** |
| **"label": "delete"** |
| **},** |
| **{** |
| **"value": "create-branch",** |
| **"label": "create a new branch and apply stash",** |
| **"usage": "git stash branch <branch name> <stash id>"** |
| **}** |
| **]** |
| **}** |
| **}** |

#### MainActivity.kt

|  |
| --- |
| **// MainActivity.kt** |
| **package com.example.githelp** |
|  |
| **import android.annotation.SuppressLint** |
| **import android.app.Activity** |
| **import android.content.Context** |
| **import android.os.Bundle** |
| **import android.text.Html** |
| **import android.view.View** |
| **import android.view.inputmethod.InputMethodManager** |
| **import android.widget.ArrayAdapter** |
| **import androidx.databinding.DataBindingUtil** |
| **import com.example.githelp.databinding.ActivityMainBinding** |
|  |
| **class MainActivity : Activity() {** |
|  |
| **private lateinit var dataBind: ActivityMainBinding** |
| **private var primaryOptions = ArrayList<PrimaryOptions>()** |
| **private var primaryOptionsValue = ""** |
| **private var secondaryOptions = ArrayList<SecondaryOptions>()** |

|  |
| --- |
| **private lateinit var dbHelper: DatabaseHelper** |
| **private var usage = ""** |
| **private var note = ""** |
|  |
| **@SuppressLint("ClickableViewAccessibility")** |
| **override fun onCreate(savedInstanceState: Bundle?) {** |
| **super.onCreate(savedInstanceState)** |
| **dataBind = DataBindingUtil.setContentView(this, R.layout.*activity\_main*)** |
| **dataBind.textGitCommand.*text* =** |
| **Html.fromHtml(*resources*.getString(R.string.*git\_command\_explorer*))** |
|  |
| **dbHelper = DatabaseHelper(this)** |
| **loadPrimaryOptionsFromDB()** |
|  |
| **dataBind.inputFirstField.setOnTouchListener { \_, \_ ->** |
| **dataBind.inputFirstField.showDropDown()** |
| **false** |
| **}** |
| **dataBind.inputSecondField.setOnTouchListener { \_, \_ ->** |
| **dataBind.inputSecondField.showDropDown()** |
| **false** |
| **}** |
|  |
| **dataBind.inputFirstField.setOnItemClickListener { \_, \_, position, \_ ->** |
| **primaryOptionsValue = primaryOptions[position].value** |
| ***dismissKeyboard*(dataBind.inputFirstField)** |
| **dataBind.cardViewSecondField.*visibility* = View.*VISIBLE*** |
| **dataBind.textNote.*visibility* = View.*GONE*** |
| **dataBind.cardViewNote.*visibility* = View.*GONE*** |
| **dataBind.inputSecondField.*text*.clear()** |

|  |
| --- |
| **dataBind.textDisplayGitCommand.*text* = ""** |
| **dataBind.textDisplayNote.*text* = ""** |
| **loadSecondaryOptionsFromDB(primaryOptionsValue)** |
| **}** |
|  |
| **dataBind.inputSecondField.setOnItemClickListener { \_, \_, position, \_** |
| **->** |
| ***dismissKeyboard*(dataBind.inputSecondField)** |
| **val selectedSecondaryOption = secondaryOptions[position]** |
| **usage = selectedSecondaryOption.usage** |
| **note = selectedSecondaryOption.nb** |
|  |
| **dataBind.textNote.*visibility* = if (note.*isEmpty*()) View.*GONE* else** |
| **View.*VISIBLE*** |
| **dataBind.cardViewNote.*visibility* = if (note.*isEmpty*()) View.*GONE*** |
| **else View.*VISIBLE*** |
| **dataBind.textDisplayGitCommand.*text* = usage** |
| **dataBind.textDisplayNote.*text* = note** |
| **}** |
| **}** |
|  |
| **private fun loadPrimaryOptionsFromDB() {** |
| **primaryOptions.clear()** |
| **primaryOptions.addAll(dbHelper.getPrimaryOptions())** |
| **val adapter = ArrayAdapter(** |
| **this,** |
| **android.R.layout.*simple\_list\_item\_1*,** |
| **primaryOptions.*map* { it.label }** |
| **)** |
| **dataBind.inputFirstField.setAdapter(adapter)** |
| **}** |
|  |
| **private fun loadSecondaryOptionsFromDB(primaryValue: String) {** |
| **secondaryOptions.clear()** |
| **secondaryOptions.addAll(dbHelper.getSecondaryOptions(primaryValue))** |
| **val adapter = ArrayAdapter(** |
| **this,** |
| **android.R.layout.*simple\_list\_item\_1*,** |
| **secondaryOptions.*map* { it.label }** |
| **) dataBind.inputSecondField.setAdapter(adapter)** |
| **}** |
|  |
| **private fun Context.dismissKeyboard(view: View?) {** |
| **view?.*let* {** |
| **val imm = getSystemService(Context.*INPUT\_METHOD\_SERVICE*) as** |
| **InputMethodManager** |
| **imm.hideSoftInputFromWindow(it.*windowToken*, 0)** |
| **}** |
| **}** |
|  |
| **override fun onDestroy() {** |
| **dbHelper.close()** |
| **super.onDestroy()** |
| **}** |
| **}** |

#### SplashScreen.kt

|  |
| --- |
| **package com.example.githelp** |
|  |
| **import android.annotation.SuppressLint** |
| **import android.app.Activity** |
| **import android.content.Intent** |
| **import android.os.Bundle** |
| **import android.os.Handler** |
| **import android.text.Html** |
| **import androidx.appcompat.app.AppCompatActivity** |
| **import androidx.databinding.DataBindingUtil** |
| **import com.example.githelp.databinding.ActivitySplashScreenBinding** |
|  |
| **@SuppressLint("CustomSplashScreen") class SplashScreen : Activity() {** |
|  |
| **companion object {** |
| **private const val DELAY\_TIME\_IN\_MILLS = 2500L** |
| **}** |
|  |
| **private lateinit var dataBind: ActivitySplashScreenBinding** |
|  |
| **override fun onCreate(savedInstanceState: Bundle?) {** |
| **super.onCreate(savedInstanceState)** |
| **dataBind = DataBindingUtil.setContentView(this,** |
| **R.layout.*activity\_splash\_screen*)** |
| **dataBind.textGitCommand.*text* =** |
| **Html.fromHtml(*resources*.getString(R.string.*git\_command\_explorer*))** |
| **Handler().postDelayed({** |
| **val intent = Intent(this, MainActivity::class.*java*)** |
| **startActivity(intent)** |
| **finish()** |
| **}, DELAY\_TIME\_IN\_MILLS)** |
| **}** |
| **}** |

#### DatabaseHelper.kt

|  |
| --- |
| **// DatabaseHelper.kt** |
| **package com.example.githelp** |
|  |
| **import android.content.ContentValues** |
| **import android.content.Context** |
| **import android.database.Cursor** |
| **import android.database.sqlite.SQLiteDatabase** |
| **import android.database.sqlite.SQLiteOpenHelper** |
| **import android.util.Log** |
| **import org.json.JSONObject** |
|  |
| **// Extension function to load JSON from assets** |
| **private fun Context.loadJSONFromAsset(filename: String): String? { // Made** |
| **'private'** |
| **return try {** |
| **assets.open(filename).bufferedReader().use { it.readText() }** |
| **} catch (e: Exception) {** |
| **e.printStackTrace()** |
| **return null // Explicitly return null in case of error** |
| **}** |
| **}** |
|  |
| **class DatabaseHelper(private val context: Context) : // context as a private** |
| **val** |
| **SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {** |
|  |
| **companion object {** |
| **private const val DATABASE\_NAME = "GitCommandExplorer.db"** |
| **private const val DATABASE\_VERSION = 1** |
|  |

|  |
| --- |
| **// Table Names** |
| **private const val TABLE\_PRIMARY\_OPTIONS = "primary\_options"** |
| **private const val TABLE\_SECONDARY\_OPTIONS = "secondary\_options"** |
|  |
| **// Primary Options Table Columns** |
| **private const val COL\_PRIMARY\_VALUE = "value"** |
| **private const val COL\_PRIMARY\_LABEL = "label"** |
| **// Secondary Options Table Columns** |
| **private const val COL\_SECONDARY\_VALUE = "value"** |
| **private const val COL\_SECONDARY\_LABEL = "label"** |
| **private const val COL\_SECONDARY\_USAGE = "usage"** |
| **private const val COL\_SECONDARY\_NB = "nb"** |
| **private const val COL\_SECONDARY\_PRIMARY\_VALUE = "primary\_value" //** |
| **Foreign Key** |
| **}** |
|  |
| **override fun onCreate(db: SQLiteDatabase?) {** |
| **// Create Primary Options Table** |
| **val createPrimaryTable = """** |
| **CREATE TABLE $TABLE\_PRIMARY\_OPTIONS (** |
| **$COL\_PRIMARY\_VALUE TEXT PRIMARY KEY,** |
| **$COL\_PRIMARY\_LABEL TEXT** |
| **)** |
| **""".trimIndent()** |
| **db?.execSQL(createPrimaryTable)** |
|  |
| **// Create Secondary Options Table** |
| **val createSecondaryTable = """** |
| **CREATE TABLE $TABLE\_SECONDARY\_OPTIONS (** |
| **$COL\_SECONDARY\_VALUE TEXT PRIMARY KEY,** |

|  |
| --- |
| **$COL\_SECONDARY\_LABEL TEXT,** |
| **$COL\_SECONDARY\_USAGE TEXT,** |
| **$COL\_SECONDARY\_NB TEXT,** |
| **$COL\_SECONDARY\_PRIMARY\_VALUE TEXT,** |
| **FOREIGN KEY ($COL\_SECONDARY\_PRIMARY\_VALUE)** |
| **REFERENCES $TABLE\_PRIMARY\_OPTIONS($COL\_PRIMARY\_VALUE)** |
| **)** |
| **""".trimIndent() db?.execSQL(createSecondaryTable)** |
|  |
| **// \*\* POPULATING DATABASE FROM JSON \*\*** |
| **// populateDatabase(db)** |
| **// Log.i("DB\_INIT", "Database populated on create.")** |
| **}** |
|  |
| **override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:** |
| **Int) {** |
| **// Handle database upgrades by dropping existing tables** |
| **db?.execSQL("DROP TABLE IF EXISTS $TABLE\_SECONDARY\_OPTIONS")** |
| **db?.execSQL("DROP TABLE IF EXISTS $TABLE\_PRIMARY\_OPTIONS")** |
| **onCreate(db) // Re-create and re-populate on upgrade (for simplicity)** |
| **}** |
|  |
| **private fun populateDatabase(db: SQLiteDatabase?) {** |
| **try {** |
| **val jsonString =** |
| **context.loadJSONFromAsset("git\_command\_explorer.json")** |
| **jsonString?.let { jsonStringNonNull -> // Renamed 'it' to** |
| **'jsonStringNonNull'** |
| **val json = JSONObject(jsonStringNonNull)** |
| **val primaryOptions = json.getJSONArray("primary\_options")** |
|  |

|  |
| --- |
| **for (i in 0 until primaryOptions.length()) {** |
| **val primary = primaryOptions.getJSONObject(i)** |
| **val values = ContentValues().apply {** |
| **put(COL\_PRIMARY\_VALUE, primary.getString("value"))** |
| **put(COL\_PRIMARY\_LABEL, primary.getString("label"))** |
| **}** |
| **db?.insert(TABLE\_PRIMARY\_OPTIONS, null, values)** |
| **}** |
| **val secondaryOptions = json.getJSONObject("secondary\_options")** |
| **val primaryKeys = secondaryOptions.keys()** |
| **while (primaryKeys.hasNext()) {** |
| **val primaryKey = primaryKeys.next() as String** |
| **val secondaryArray =** |
| **secondaryOptions.getJSONArray(primaryKey)** |
| **for (i in 0 until secondaryArray.length()) {** |
| **val secondary = secondaryArray.getJSONObject(i)** |
| **val values = ContentValues().apply {** |
| **put(COL\_SECONDARY\_VALUE,** |
| **secondary.getString("value"))** |
| **put(COL\_SECONDARY\_LABEL,** |
| **secondary.getString("label"))** |
| **if (secondary.has("usage")) {** |
| **put(COL\_SECONDARY\_USAGE,** |
| **secondary.getString("usage"))** |
| **}** |
| **if (secondary.has("nb")) {** |
| **put(COL\_SECONDARY\_NB,** |
| **secondary.getString("nb"))** |
| **}** |
| **put(COL\_SECONDARY\_PRIMARY\_VALUE, primaryKey)** |
| **}** |
| **db?.insert(TABLE\_SECONDARY\_OPTIONS, null, values)** |

|  |
| --- |
| **}** |
| **}** |
| **} ?: run {** |
| **Log.e("DB\_INIT", "Error loading JSON for population.")** |
| **}** |
| **} catch (e: Exception) {** |
| **Log.e("DB\_INIT", "Error populating database:** |
| **${e.localizedMessage}")** |
| **}** |
| **}** |
|  |
| **// \*\* The app will use these methods to retrieve data from the database \*\*** |
| **fun getPrimaryOptions(): List<PrimaryOptions> {** |
| **val primaryOptionsList = mutableListOf<PrimaryOptions>()** |
| **val db = readableDatabase** |
| **val cursor: Cursor = db.query(** |
| **TABLE\_PRIMARY\_OPTIONS,** |
| **arrayOf(COL\_PRIMARY\_VALUE, COL\_PRIMARY\_LABEL),** |
| **null, null, null, null, null** |
| **)** |
| **cursor.use {** |
| **while (it.moveToNext()) {** |
| **primaryOptionsList.add(** |
| **PrimaryOptions(** |
|  |
| **it.getString(it.getColumnIndexOrThrow(COL\_PRIMARY\_VALUE)),** |
|  |
| **it.getString(it.getColumnIndexOrThrow(COL\_PRIMARY\_LABEL))** |
| **)** |
| **)** |
| **}** |
| **}** |

|  |
| --- |
| **db.close()** |
| **return primaryOptionsList** |
| **}** |
|  |
| **fun getSecondaryOptions(primaryValue: String): List<SecondaryOptions> {** |
| **val secondaryOptionsList = mutableListOf<SecondaryOptions>()** |
| **val db = readableDatabase** |
| **val cursor: Cursor = db.query( TABLE\_SECONDARY\_OPTIONS,** |
| **arrayOf(** |
| **COL\_SECONDARY\_VALUE,** |
| **COL\_SECONDARY\_LABEL,** |
| **COL\_SECONDARY\_USAGE,** |
| **COL\_SECONDARY\_NB** |
| **),** |
| **"$COL\_SECONDARY\_PRIMARY\_VALUE = ?",** |
| **arrayOf(primaryValue),** |
| **null, null, null** |
| **)** |
| **cursor.use {** |
| **while (it.moveToNext()) {** |
| **secondaryOptionsList.add(** |
| **SecondaryOptions(** |
|  |
| **it.getString(it.getColumnIndexOrThrow(COL\_SECONDARY\_VALUE)),** |
|  |
| **it.getString(it.getColumnIndexOrThrow(COL\_SECONDARY\_LABEL)),** |
|  |
| **it.getString(it.getColumnIndexOrThrow(COL\_SECONDARY\_USAGE)) ?: "",** |
|  |
| **it.getString(it.getColumnIndexOrThrow(COL\_SECONDARY\_NB)) ?: ""** |
| **)** |
| **)** |
| **}** |
| **}** |
| **db.close()** |
| **return secondaryOptionsList** |
| **}** |
|  |
| **override fun close() { readableDatabase.close()** |
| **writableDatabase.close()** |
| **super.close()** |
| **}** |
| **}** |

#### PrimaryOptions.kt

|  |
| --- |
| **package com.example.githelp** |
|  |
| **data class PrimaryOptions(val value: String, val label: String)** |

**SecondaryOptions.kt**

|  |
| --- |
| **package com.example.githelp** |
|  |
| **data class SecondaryOptions(** |
| **val value: String,** |
| **val label: String,** |
| **val usage: String, val nb: String** |
| **)** |