My understanding of my app and how it works...

I was assigned to create an app to help study using flash cards.

It is a history exam coming up and the app allows me to answer 5 questions in true or false format, let's take a closer look on how it works.

The primary goal of this flashcard quiz app is to provide users with an engaging, accessible, and efficient way to study historical facts through true/false questions. The app is particularly valuable for students, trivia enthusiasts, and individuals preparing for exams where historical knowledge is required.

Key Objectives:

- Reinforce historical knowledge in a game-like format.
- Support quick sessions of study using flashcards.
- Deliver a mobile-native experience through Android.

Features:

- A simple UI showing a question and true/false buttons.
- Instant feedback on the answer.
- Score tracking and progress display.
- Lightweight and responsive for use on a range of Android devices.

Designing the app involved several important considerations to ensure usability, performance, and maintainability.

User Experience (UX)

- Minimalistic interface to reduce distractions.
- Clear, large text for readability.
- Quick animations for feedback on answers.

- Easy navigation for users of all ages.

User Interface (UI)

- Utilizes Material Design components for consistency and accessibility.
- Custom color schemes to evoke a calm, focused environment.

Architecture

- Kotlin used for native Android development.
- MVVM (Model-View-ViewModel) architecture to separate UI from logic.
- Data classes used to model flashcards.
- ViewModel and LiveData to manage UI-related data in a lifecycle-conscious way.

Testing & Debugging

- Unit tests for logic such as answer checking and score tracking.
- UI tests using Espresso (planned or implemented).

GitHub serves as the central platform for version control, collaboration, and code management.

Repository Structure

- Code is organized into packages: ui, model, viewmodel, etc.
- README.md provides setup instructions and an overview.
- Branching strategy used for features, fixes, and releases.

Collaboration & Documentation

- Issues used to track tasks and bugs.
- Pull Requests reviewed and merged after CI checks.
- GitHub Discussions or Wiki (if used) to document design decisions or common Q&A.

4. GitHub Actions Integration

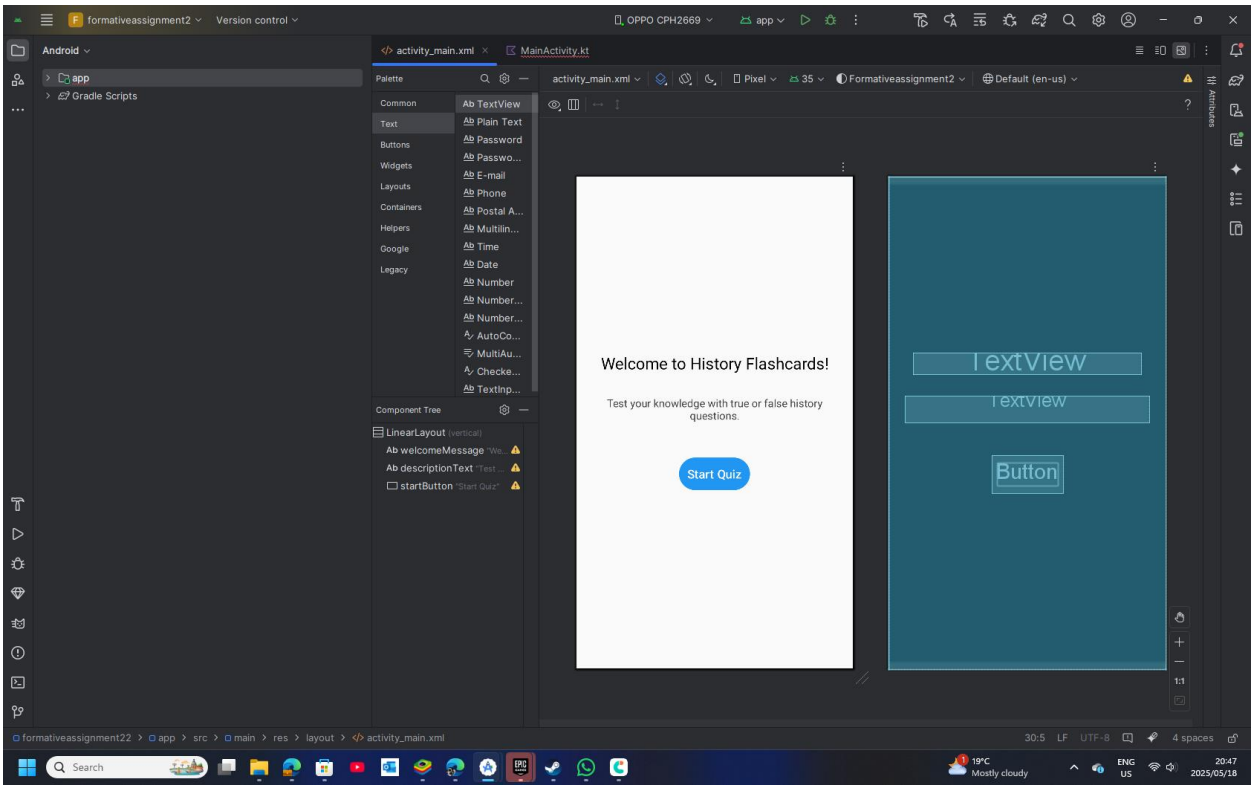
GitHub Actions automates the development workflow, ensuring code quality and consistency with every commit or pull request.

CI/CD Workflows

- CI (Continuous Integration):
- Triggered on push or pull_request.
- Builds the app using Gradle.
- Runs unit tests automatically.
- Linting & Code Quality:
- Android Lint and ktlint used to enforce coding standards.
- Artifacts & Reports:
- Workflow uploads build artifacts (APK or AAB) for testing.
- Test results are summarized in PR comments or logs.

Benefits:

- Early detection of build or test failures.
- Faster feedback loop during development.
- Confidence in code quality before merging.



Reference:

. Android Development

Android Developers, 2024. Build your first app. [online] Android Developers. Available at: https://developer.android.com/training/basics/firstapp

2. Kotlin Programming Language

JetBrains, 2024. Kotlin documentation. [online] Available at: https://kotlinlang.org/docs/home.html

3. MVVM Architecture

Google Developers, 2024. Guide to app architecture. [online] Android Developers. Available at: https://developer.android.com/topic/architecture .

4. GitHub Version Control

GitHub, 2024. GitHub Docs: About version control. [online] Available at: https://docs.github.com/en/get-started/using-git/about-version-control

5. GitHub Actions (CI/CD)

GitHub, 2024. Understanding GitHub Actions. [online] Available at: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions

6. Material Design Guidelines

Google, 2024. Material Design. [online] Available at: https://m3.material.io/