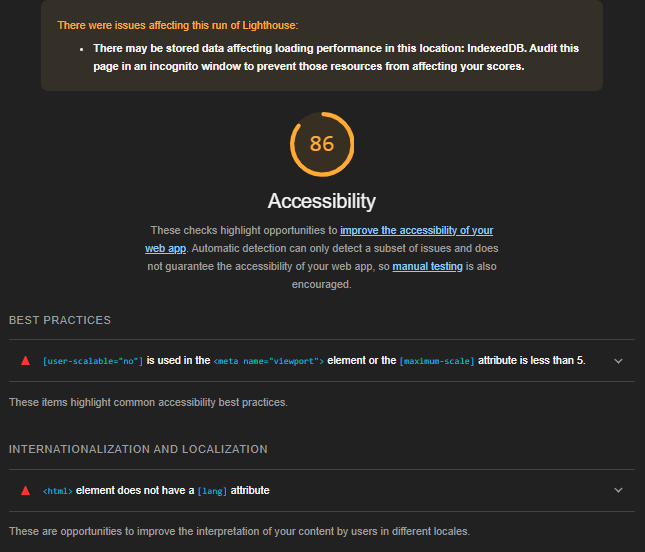
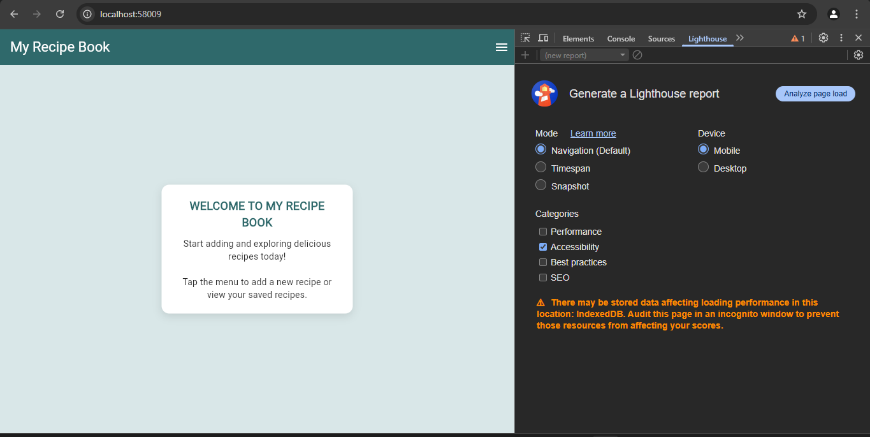
1. Accessibility Evaluation (R&U1 – 2 Marks)

To assess the accessibility of my Flutter application, I used **Google Lighthouse**, a built-in auditing tool available in Chrome DevTools.

**Results:**

The accessibility test returned a score of **86/100**, indicating that the application meets most accessibility requirements but has some areas for improvement.

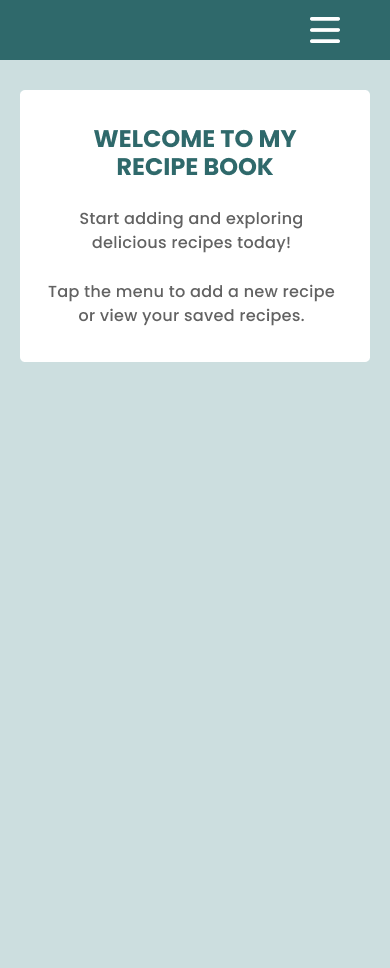
**Identified Issues:**

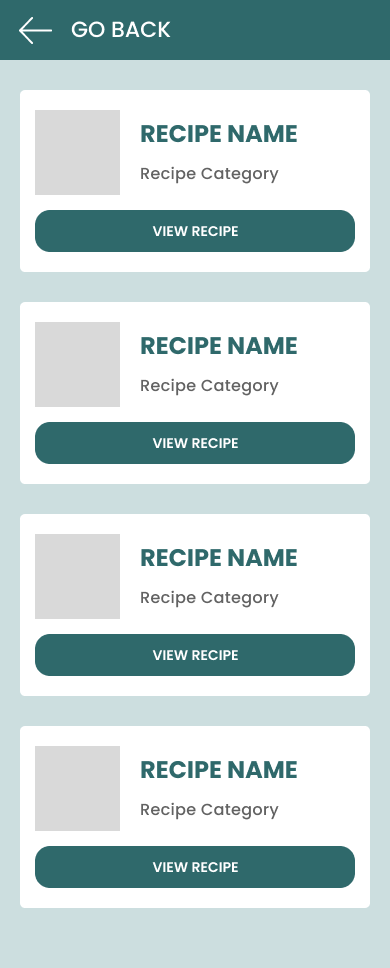
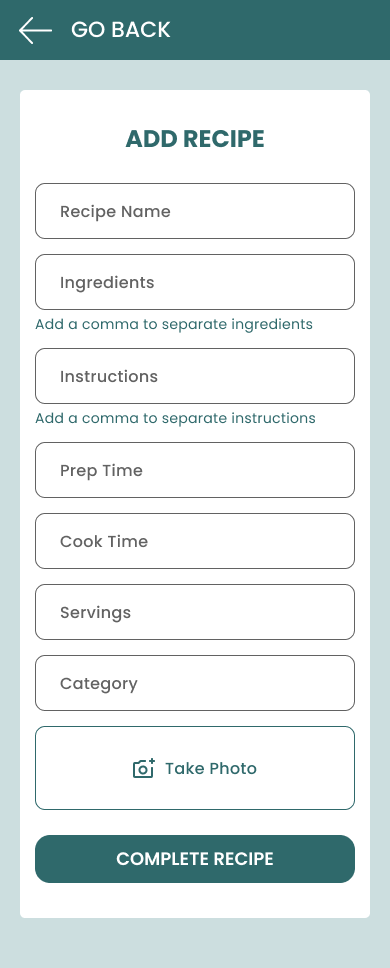
* + The <meta name="viewport"> tag includes user-scalable="no", which restricts users from zooming in. This can negatively impact accessibility for visually impaired users.
  + The <html> element is missing a lang attribute, which can affect screen readers and other assistive technologies.

1. Future Enhancements (R&U7 - 8 Marks).

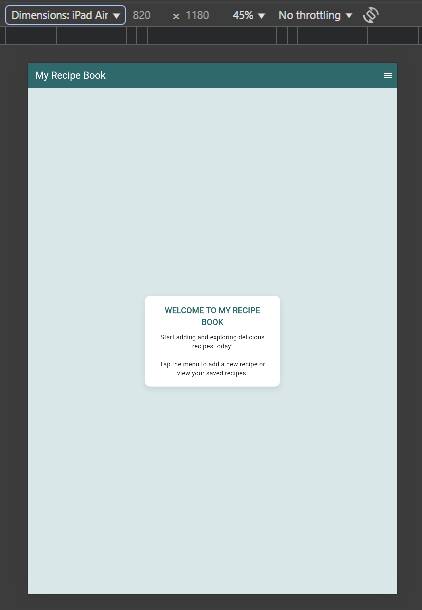
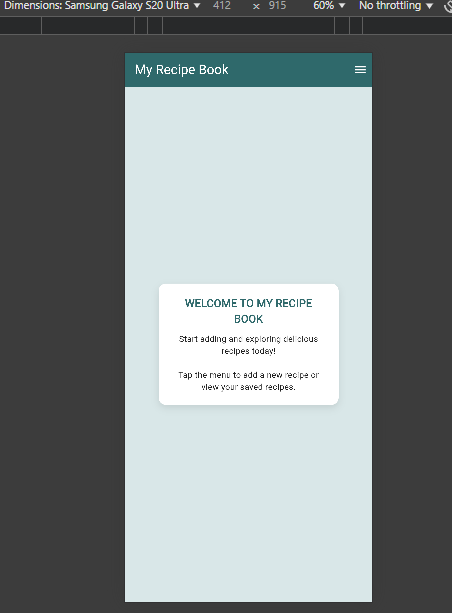
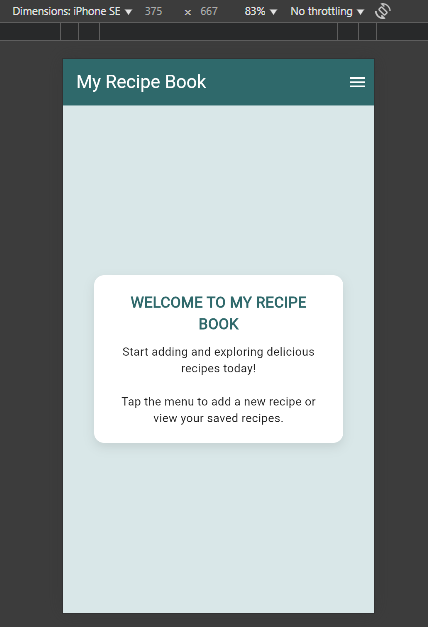
* **Search & Filtering:** Add a search bar and category filters to help users quickly find recipes by ingredients, cooking time, or name.
* **User Authentication:** Implement Firebase Authentication to allow users to create accounts and save their own recipes securely.
* **Recipe Sharing:** Introduce a "Share" button for users to easily share recipes via social media, WhatsApp, and email, enhancing engagement.

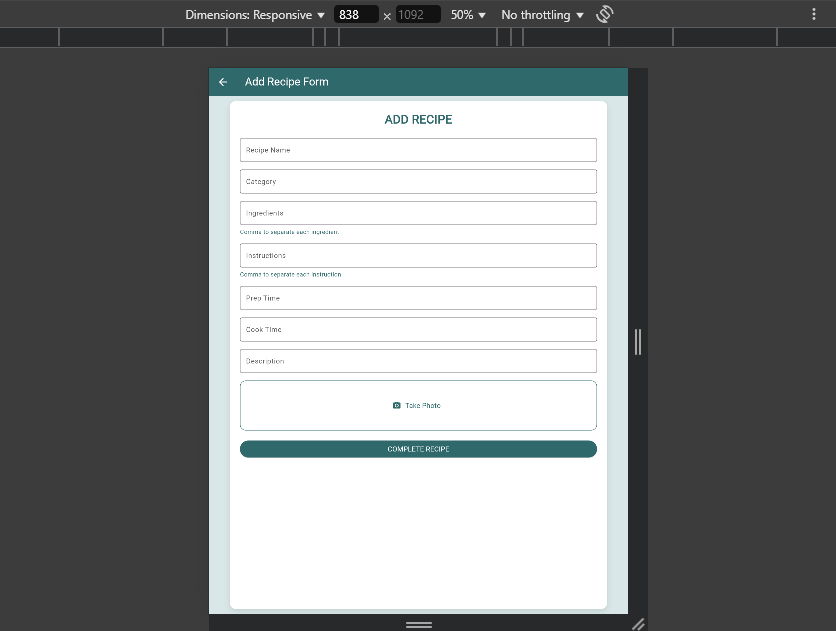
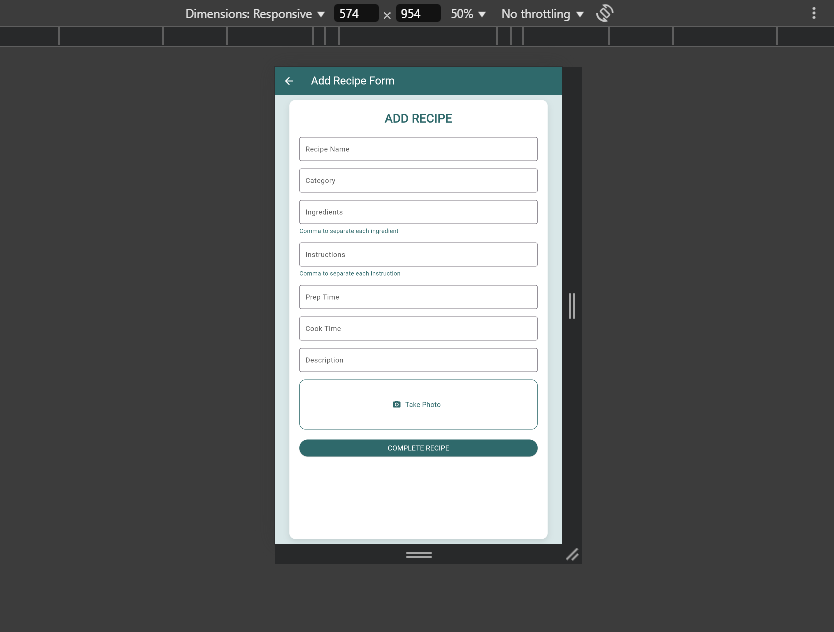
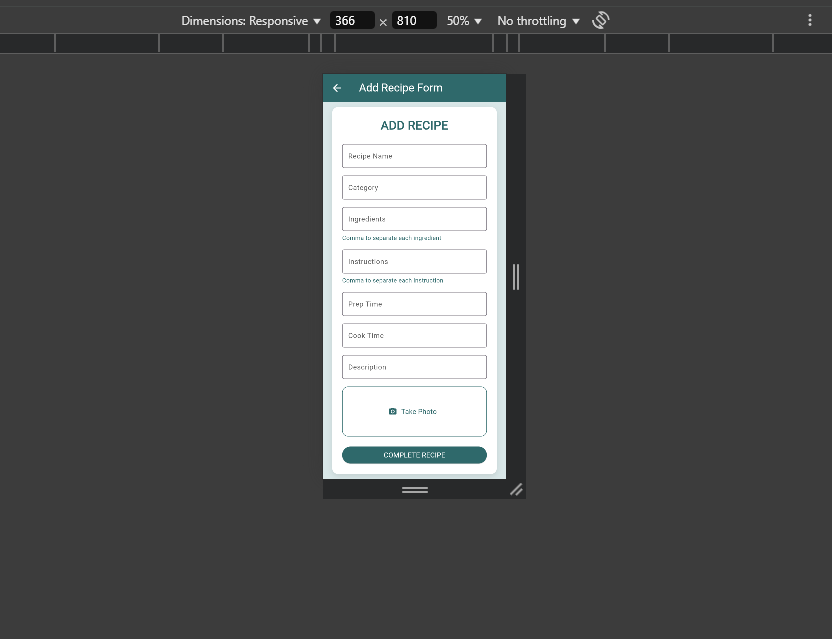
1. Design & Prototyping (E&C2 – 8 Marks)





1. Responsive Design Screenshots (A&A1 – 3 Marks).

Small(375x667) Medium(412x915) Large(820x1180)



1. Performance Analysis(R&U2, E&C3)

To evaluate the UI performance of my Flutter application **“My Recipe Book”**, I intended to use **Flutter DevTools**, which provides a suite of performance profiling tools such as:

* **Performance view**: for frame rendering, build and layout tracking.
* **Timeline view**: to identify slow frames and jank.
* **Memory view**: to detect leaks or excessive allocation.
* **CPU Profiler**: for analysing Dart and native code execution.

However, **Flutter DevTools was unable to detect my application when running**, despite multiple attempts to reconnect and refresh. As a result, I used **Chrome DevTools in profile mode** to inspect rendering performance and input delay.

**Tools Used:**

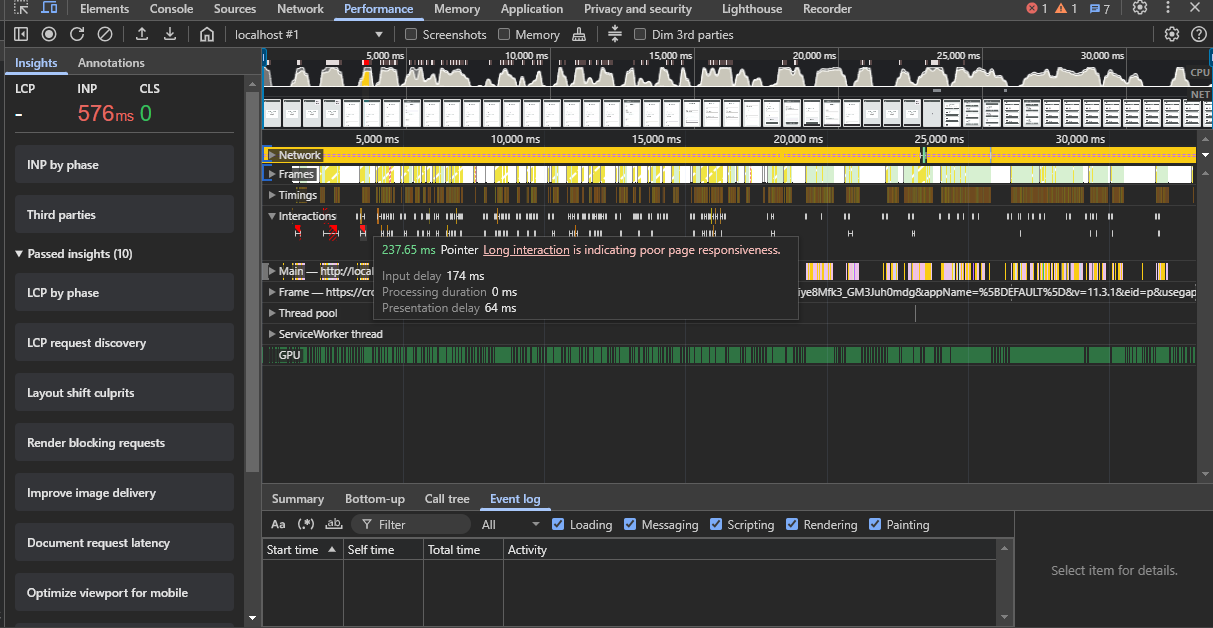
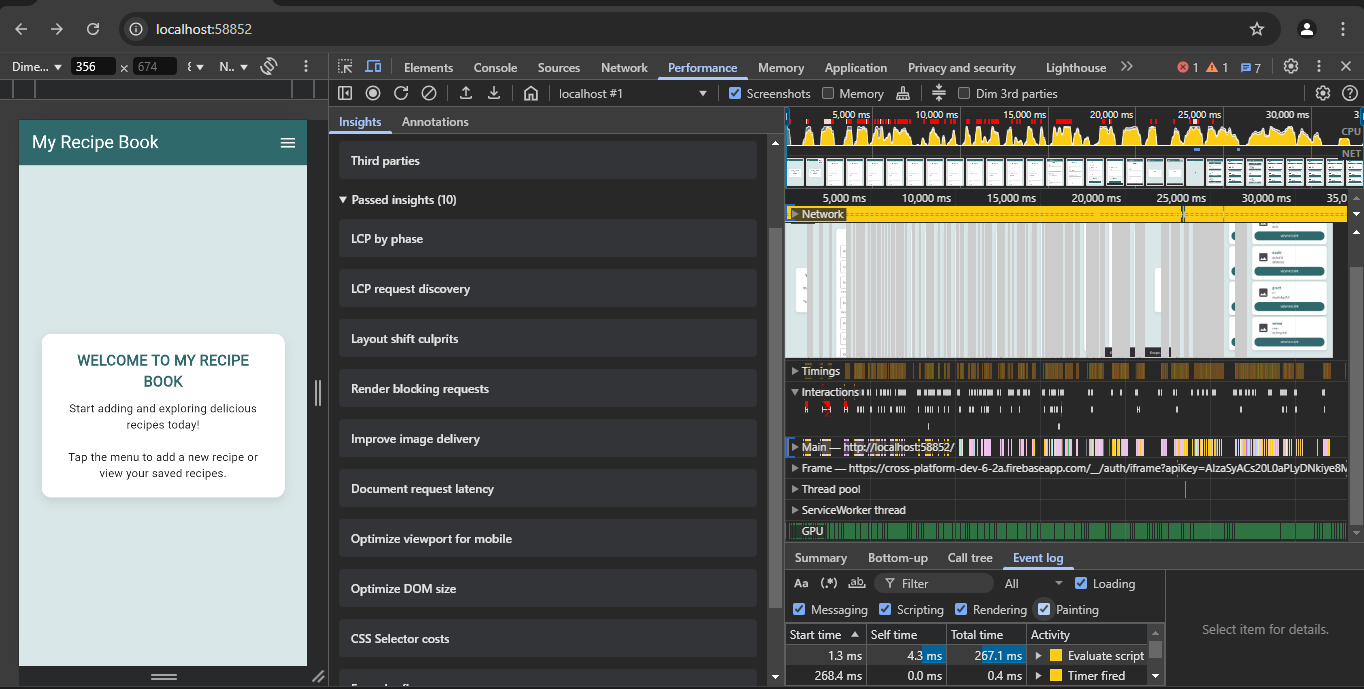
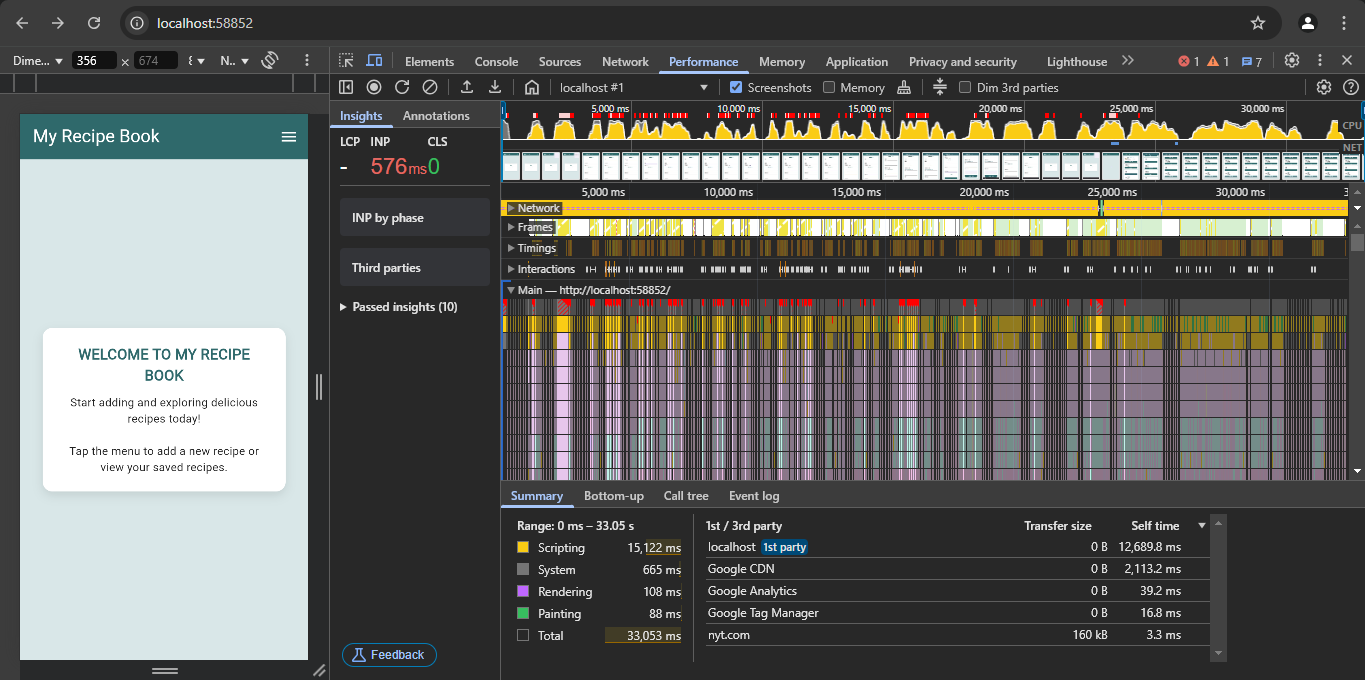
* **Chrome DevTools, Performance tab**

**Performance Insights:**

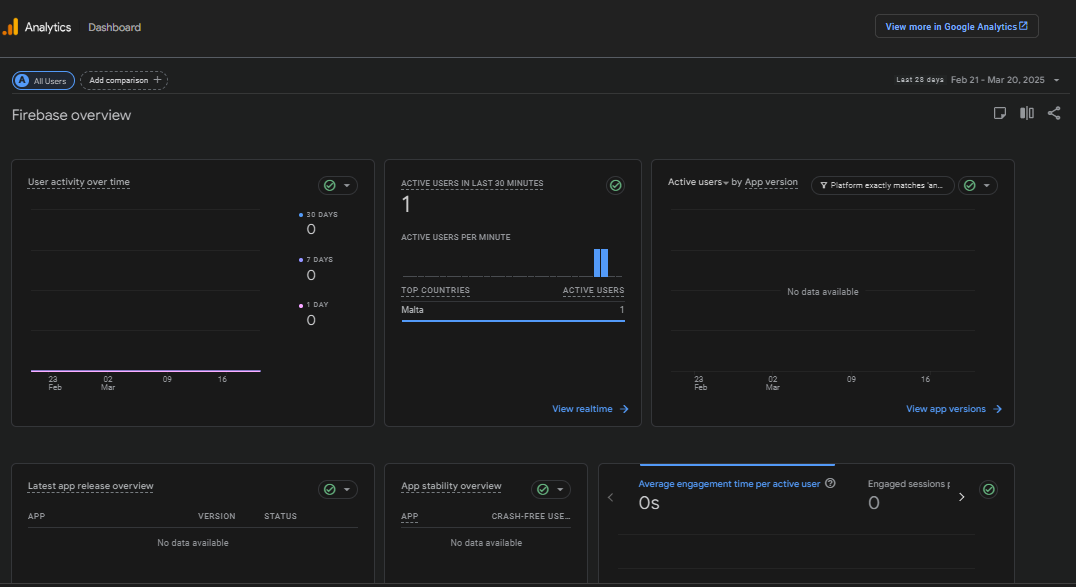
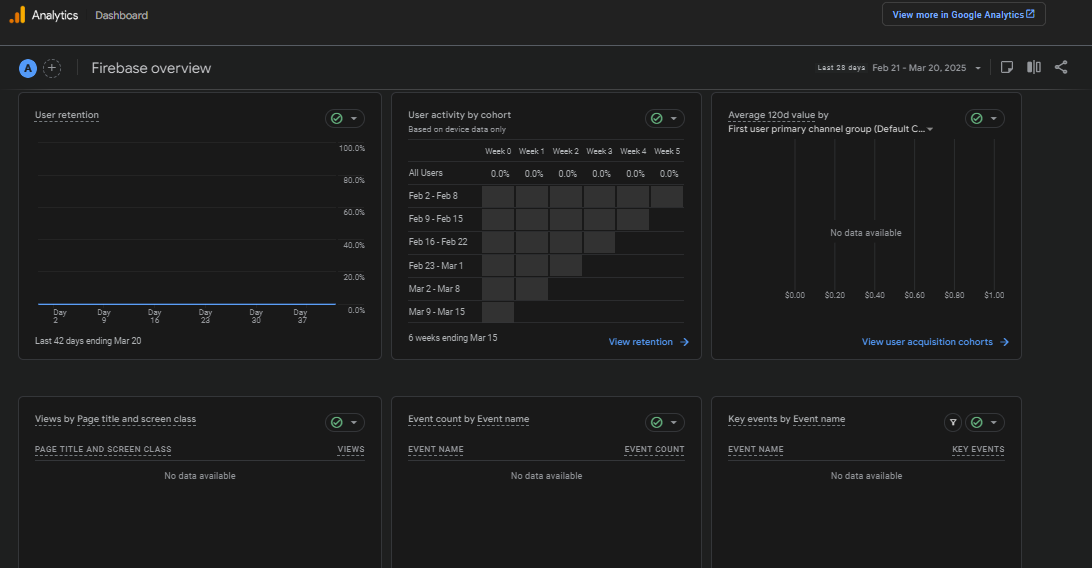
* **Total Runtime**: ~33 seconds recorded session
* **Scripting time**: ~15,122ms
* **Rendering**: 108ms
* **Painting**: 88ms
* **INP (Interaction to Next Paint)**: **576ms**, which is higher than ideal (under 200ms is recommended)
* **Long Interaction Warning**: Detected at ~237ms pointer delay, suggesting occasional UI jank

**Reflections:**

Although Chrome DevTools is not designed specifically for Flutter apps, it proved useful for analysing **frame activity**, **scripting delays**, and **UI responsiveness**. Based on the data, I can work on **optimizing state changes**, **reducing script-heavy operations**, and improving overall render performance. Once Flutter DevTools is accessible, I plan to re-profile the app using its native tools for deeper Dart-level insights.



1. Firebase Analytics Documentation (R&U6 – 8 Marks)

Firebase Analytics allows mobile applications to collect a **wide range of analytics data** to monitor **user behavior, app performance, and engagement**. These analytics are categorized into **automatically collected events** and **custom events.**

### ****User Engagement Metrics****

* **Active Users** → Tracks the number of users currently using the app.
* **Session Duration** → Measures how long users spend in the app per session.
* **Screen Views** → Logs which screens users visit the most.
* **User Retention** → Tracks how often users return to the app..

### ****Event-Based Analytics****

* **Button Clicks** → Tracks when users interact with UI elements.
* **Form Submissions** → Logs when users enter and submit data.
* **Search Queries** → Captures what users search for in the app.
* **Recipe Additions (Custom Event)** → Can be set up to track when users add new recipes.

### ****App Performance & Stability****

* **Crash Reports** → Identifies when and why the app crashes.
* **App Load Time** → Measures how quickly the app starts up.
* **Network Request Performance** → Tracks response times for API requests.

1. Description of GitHub Actions and CI/CD (6 Marks)

GitHub Actions is a tool that automates tasks like testing and building your Flutter app. It is used in CI/CD (Continuous Integration and Continuous Deployment) to ensure the app works correctly every time code is pushed or updated.

In this project, GitHub Actions can:

* Check for code issues using flutter analyze
* Run tests with flutter test
* Build the app with flutter build

This helps keep the app reliable and ready to deploy at any time.

1. Code with description demonstrating how GitHub Actions can be integrated (2 Marks)

name: Flutter CI

on: [push, pull\_request]

jobs:

  build:

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v3

      - uses: subosito/flutter-action@v2

        with:

          flutter-version: '3.13.0'

      - run: flutter pub get

      - run: flutter analyze

      - run: flutter test

      - run: flutter build web

      - run: flutter build apk

      - run: flutter build ios

* **Triggers**: Runs when code is pushed or a pull request is made.
* **Steps**:
  1. **Check out the latest code**
  2. **Set up Flutter**
  3. **Install dependencies**
  4. **Analyze code & run tests**
  5. **Build the app for deployment**

This ensures that every update is **tested and verified** before merging.

### iOS Notification Requirements (R&U4 – 4 marks)

To implement push notifications on iOS, the app must request permission using UNUserNotificationCenter. It also requires setting up notification capabilities in Xcode and configuring APNs (Apple Push Notification Service). In production, a valid Apple Developer account and provisioning profile are also required.