Software Project
Assignment 5: Quality Attributes and Architecture

Project: TourGuide Manager

Customers: **Karina Shavalieva** (@K_Arbyzova), **Hannanov Rishat** (@RishatHannanov)

Team 59 (Members and Their Contribution):

- **Alexey Chegaev** (wyroxx) - Team lead, configure database and doing some fixes.
- **Aleksandr Medvedev** (BearAx) - flutter developer, programming calendar for guides, doing some fixes.
- **Nikita Shankin** (Mysteri0K1ng) - flutter developer, programming profile, doing some fixes.
- **Georgii Beliaev** (JoraXD) - web-developer, writing web-site for admins.
- **Alexander Simonov** (AlexbittIT) - web-developer, writing blacklist of users and guides.

Date: 06.07.2025

Links

- **GitHub (Main Project):** Link
- **GitHub (Product Backlog):** Link
- **GitHub (Milestone):** Link
- **Story Points Tracking Sheet:** Link
- **Product Roadmap:** Link
- **The Video Demo:** Link
- **Deployed Improved Product:**
  - Website (for Guides): Link
  - Website (for Admins): Link
- **Tests:** Link
- **Integration Tests (cannot be run, there is no emulator for them in the github):** Link
- **All Processes:** Link

Retrospective on the process and customer collaboration

We already have 2 ready-made sites linked to 1 database (firebase), so next (after MVP 2) we plan to make some fixes and clarify with the customer exactly how notifications should work, as well as whether there is a need to add something else, we will also optimize the code.

CD / CI

GitHub Actions CI for Flutter

- **File:** `.github/workflows/flutter_ci.yml`
- **Link**

What it does

- **Runs tests**
- **Pull requests targeting** `main`

Steps included

- **Checkout the repository**
  - **Uses** `actions/checkout@v4`
- **Install Flutter**
  - **Uses** `subosito/flutter-action@v2` **with:**
    - **Flutter version:** `3.32.0`
    - **Channel:** `stable`
    - **Caching enabled**

Steps included

- **Verify Flutter version**
  - **Runs** `flutter --version`
- **Install project dependencies**
  - **Runs** `flutter pub get`
- **Analyze the code (linting)**
  - **Runs** `flutter analyze`
- **Run unit tests**
  - **Runs** `flutter test`
- **Run integration tests (optional)**
  - **Runs:**

```
set +e
flutter test integration_test || true
```

Purpose

- **This pipeline ensures code quality and stability by automatically analyzing the code and running tests whenever changes are pushed or merged into the main branch.**

Justified List of QA Tools Used

- `flutter analyze`
  - Used to perform static code analysis. It helps identify syntax errors, unused imports, and code that doesn't follow Dart or Flutter best practices. This ensures cleaner, more maintainable code across the project.
- `flutter test`
  - Used to run unit tests on business logic and smaller isolated components of the application. It helps verify that individual functions or classes work correctly.
- `integration_test`
  - Used to perform integration testing by simulating real user interactions with the app (e.g., tapping buttons, filling forms). This validates that different components work together as expected.

Justified List of QA Tools Used

- `flutter_lints`
  - A predefined set of linting rules recommended by the Flutter team. It enforces consistent coding style and best practices across the team, reducing bugs caused by inconsistent code.
- `GitHub Actions`
  - Used to automate CI/CD. It runs all the checks (analyze, test) automatically on each push or pull request to the main branch. This ensures code quality is continuously maintained and regressions are caught early.
- `flutter test --coverage`
  - Generates test coverage reports to track how much of the code is exercised by tests. This helps identify untested code areas and improve test completeness.

AI Usage: We used AI when working with github, as well as when creating diagrams.

Screenshots of testing reports