Test Plan for Tutor-Me

Scope

This project focuses on providing a platform to ease the process of:

- Registering as a Tutor that can tutor certain modules for a price,
- Registering as a Tutee that needs a tutor for certain modules,
- Allowing the tutor to select the modules he/she tutors for a fee,
- Allowing the tutee to choose which subject(s) they need help with from certain tutors,
- One-on-one chat functionality between Tutees and Tutors,
- Group chat creation for students requiring assistance with the same modules or subjects,
- Sending and receiving notifications when Tutor is found
- Video calling functionality in-app to allow students to virtually meet
- After implementing these core functionalities of this project, we would start looking at additional functionalities like filtering tutors by gender to enable students to choose someone that they are comfortable with. We would also look into implementing a dark mode of the app for users that prefer it over light mode. The project will be implemented as a cross platform mobile app for Android and iOS.

Environment

automated tests on github

Tools of running the tests

- Run “flutter test” or” pub run test” inside the /test folder
- Run “ flutter analyze” before commit/pull request on github ,so that it can perform static analysis to the code.

Defect Management

- Project board for reporting

Risk Management

- Running out of time and feature not implemented
- Bug ridden feature implemented
- Incoherent design for following group
- Project Owners not agreeing with feature analysis
- Feature is poorly designed
- Insecure feature

Test Type

- **Unit testing**
  Testing Individual modules in isolation to make sure it works correctly.
- **Widget Testing (Ui Testing)**
  Testing whether the widget works as expected.
- **Integration testing**
  Testing (Individual modules are combined and tested as a group)
- **User acceptance testing**
  Performed by the end user or client to verify or accept the software system before it is moved into the production environment.

Test Logistics

- **Who will test?**
  - developers and the tester
- **When will the test occur?**
  - Test Specification is created
  - Test Environment is built

TEST OBJECTIVE

- To determine practical storage limitations before deployment.
- To determine behavior of the system when a new hardware device is replaced or any existing device is upgraded.
- To minimize the response time.
- Do many unit and widget tests, plus enough integration tests to cover all the important use cases.

TEST CRITERIA

Testing Protocols

Standard Format to be followed:

★ **Pass**: Date, Description, Developer, Tester, Commit hash, test status (Pass/Fail)

★ **Fail**: Date, Description, Details of the Error, Developer, Tester, Commit hash, test status (Pass/Fail)

TEST DELIVERABLES

- Tests and testing logs (including unit tests) will be stored in a subdirectory of the repository with the same standard format as the libs.
- All tests listed above should passed before PR is merged

Testing of our 3 main Non functional Requirements:

1. Security

Security is an essential part of our system as users are required to upload files and confidential documents that can be detrimental if leaked. So to ensure secure storage and retrieval of these documents and files we made use of JWT Tokens.
We then went on to implement functions that rely on the JWT Token, this ensures users who are authorized have a token to do certain functionality. Hence user authorization testing .

2. Performance

Performance is a very essential part of our system as the core functionality (Collaboration between tutors and tutees) occurs in real time. Hence it is important for data exchange to happen as quickly as possible.
To test for this we made use of a flutter performance testing method (**performance profiling**):

1. `dart devtools` . This is to ensure we have access to the developer tools offered in flutter && dart.
2. `flutter run` . To run the app. It should open a new
Window in the browser.