Team: HvZ App

Members: Kyra Clark, Santiago Rodriguez, Alex Hadley, Matthew Waddell
Project 3C.1 — Component Design

Github Link:
https://github.com/alexhad6/HvZ-app/blob/master/Administrative/Phase%203/design_3c2_sr.pdf
Primary author: Santiago

Component Test Plan: File Storage

Goals:
Test that uploading and downloading from file storage works correctly.

Testing Framework:
The testing frameworks available for this component are flutter unit testing (https://flutter.dev/docs/testing#unit-tests) and mockito (https://pub.dev/packages/mockito) . Flutter unit testing allows users to test their functions, methods, or classes within an easy to use automated testing framework. Given that the file storage component must interact with the internet, it would not be efficient to run a test and wait for what might happen with the online service. Also, any errors with the online service might interfere with the testing process. That's where mockito comes in. Mockito allows you to create a fake web service for all the intended functionality that you desire. While it would be a multi year project on its own to create firebase from scratch, mockito allows us to make a mock version of it with only the functionality we need to make sure that our functions do what we expect them to do. It's a perfect addition to unit testing because it, unlike the real firebase service, won't require the larger app infrastructure to create a testing environment. Mockito is compatible with and even recommended by the flutter unit tests.

Tests:
Upload tests:

- Make sure the file storage has stored the URL of an uploaded file in the database
    - Reason: must do this so that the file can be found again later for downloading
    - Set up: mockito version of file storage and mock addFile() database function.
    - Correctness recognition: the URL passed to the addFile() function is the same as a predetermined correct URL.