Echos

“One place to learn and translate American Sign Language”

Testing documentation

Group 11

Prayag Savsani [AU1841035]
Darshan Navadiya [AU1841061]
Kishan Mehta [AU1841072]
Harsh P. Patel [AU1841075]
Dhruv Sharma [AU1841102]
Harin Khakhi [AU1841116]
Nilay Gandhi [AU1841118]
Sanket J Shah [AU1841120]

TABLE OF CONTENTS

Introduction

This document briefs the following points:

- Discuss the testing details for each use case of our system
- The test plan which defines the testing strategies and techniques/tools used

We will also be addressing the various standards being applied to our testing suite. The documentation given below adheres to the standard testing documentation practices.

Purpose

The purpose ot the test plan is as follows,

- To describe the features being tested
- To define the entry and exit criteria for testing
- To define the pass and fail criteria for testing
- To describe the testing techniques used for testing
- To describe the deliverables of testing

Scope

Functions to be tested

User data related functions

- Creating an account
- Validating the details entered
- Signing in
- Viewing a user profile
- Updating details

ASL to text translation

- Testing the ML model
- Testing the live camera feed feature
- Validating the predicted results

Learning ASL

- Testing the video playback

- Testing the chapter-wise distribution of tutorial videos

ASL Quiz

- Testing the rendering of the quiz screen
- Validating the data fetched from the datastore
- Testing the scoring of the quiz
- Validating the generated statistics

Community section

- Testing the rendering of the community pages
- Validating the fetched data from the datastore
- Create new posts
- Comment on posts
- Liking posts

Testing approach

Given our DevOps approach, the basis of planning, estimating and designing the tests was part of the requirements gathering phase itself. The test cases were also designed during the same phase. Then we determined the testing strategy being used for each function (to be written in the coding phase). Once the coding phase started, we tested each of the functions using our pre-designed testing strategies. Hence, on every commit to the main branch or when merging a PR, our test suite was being executed.

Entry criteria

Once the requirements mentioned in entry criteria are met, the code written undergoes testing. They are given as below,

- The code satisfies the requirements criteria
- The required test code is available
- The test plan and the test cases are designed
- The test environment is setup
- The code is thoroughly reviewed and merged into the main branch

Once all these requirements are set and the code is merged to our main branch, the automated test suite is executed.

Exit criteria

Once the requirements mentioned in exit criteria are met, the code written is certified as tested. Our automated test suite takes care of validating most of these criteria. They are given as below,

- All test cases are passed
- The final build is generated without any errors during the build process

Suspension criteria

Our automated test suite suspends the currently running test process when the build process or any other preprocessing step like dependency management fails.

Passing criteria

If the tests are passing and the final build is generated without any errors, the tests are considered as passing. After this, the code is reviewed by others and if that is also positive, incoming code is merged into the main branch.

Failing criteria

Either the tests are failing or due to any changes in the build environment, the build fails to generate. These are the two cases where the tests will fail and the developer will be requested to review the code once again.

Testing strategies

Unit testing

Under this strategy, individual modules are tested and their core logic is tested. We use manual scripts for testing the modules before pushing any code. Hence, this is a manual testing technique in our test suite.

Test cases:

Test case	Test ID	Test case description	Input data (if any)	Expected output data (if any)	Test case result
Validating sign up data	1.1.1	Checking if the correct data is transferred to the database	User data	Correct data uploaded	PASS
Validating sign in data	1.2.1	Checking if the correct data is transferred to the database	User login info	Correct login info passed	PASS
User Statistics	1.4.1	Checking if the user statistics are updated on database when user completes a chapter	-	Updated statistics	PASS
Edit Profile	1.5.1	Check if the entered content is updated properly on database	Updated data	Database update	PASS
Change Password	1.6.1	Check if new password is valid	New password	-	PASS
View personal information	1.8.1	Check if correct information is fetched from the database	-	User information	PASS
Difficulty selection button working	2.1.1	Checked whether all the three buttons are working and passing desired data	-	Successful redirection to the desired page	PASS
Chapter list can be expanded	2.2.1	Checked whether the chapter list card is	-	All the chapter list card are expanding properly	PASS

		expanding properly			
Video player working properly	2.3.1	Checked whether the video player components like pause, speed up, etc	-	All the components are working perfectly.	PASS
Reading material rendered properly	2.4.1	Checked whether the data displayed is correct and does not overflow	-	All the data and widgets are rendered properly.	PASS
Correct and Incorrect answers in quiz	2.5.1	Checked whether user is properly notified upon marking correct or incorrect answer	-	All the answers are fetched from the database properly and checked with selected options. And the result is shown properly	PASS
Check score page animation	2.6.1	Checked whether the animation to display score is working properly.	-	The animation is working properly	PASS
Feedback of progress	2.7.1	Check whether the user is notified upon the completion of any component	-	User is notified upon successful completion	PASS
Posts fetching from database	3.1.1	Checking if all the posts are displayed	-	All the posts from database	PASS
Post like updated on database	3.2.1	Validating is liking/unliking posts is	-	Like status updated on database	PASS

		updated on database			
Post creation on database	3.3.1	Validating the creation of posts on database	Post title, post content	Post added on database	PASS
Comment creation on database	3.4.1	Validating the creation of comments on database	Comment content	Comment added on database	PASS
Fetch comments related to the post	3.5.1	Checking if all the posts on a given post are fetched	-	Comments from clicked post	PASS
Accessing the ML model	4.2.1	Validating whether the tflite flutter plugin returns the loaded model correctly	File Path to the saved model weights	Successfully returns the loaded model	PASS
Predicted results from the ML model	4.2.2	Validating whether the predictions from the classifier are from the valid set of classes	Live video feed from the user	Predicted class belongs the set of classes	PASS

Integration testing

Under this strategy, we test the whole application and our workflow after integrating a given module. This is again a manual testing technique.

Functional testing

Under this strategy, we test a given module of the application which represents a significant feature in our product. This is again a manual testing technique.

Test cases:

Test case	Test ID	Test case description	Priority	Expected output data (if any)	Test case result
SignUp	1.1	Check if new user created	High	Account was created successful	PASS
SignIn	1.2	User authentication works properly	High	Successfully logged in	PASS
SignOut	1.3	User signed out properly	Medium	Successfully logged out	PASS
User Statistics	1.4	User statistics displayed	Medium	Score was accurate	PASS
Edit Profile	1.5	Check if user can edit the profile	Medium	Edited profile	PASS
Change Password	1.6	Check if new password is updated properly	Medium	Updated password	PASS
Reset all progress	1.7	Check if user statistics are reset	Low	Progress reset	PASS
View Personal Information	1.8	User information displayed properly	Low	User information	PASS
Shows Proper Difficulty Selection screen	2.1	Checked whether proper corresponding titles and gifs are loaded	High	Proper titles and gifs are loading	PASS
Fetch Chapter List from database	2.2	Checked whether desired content corresponding to selected	High	Desired chapter content is fetching	PASS

		difficulty is fetched			
Fetch video from YoTube Data API	2.3	Checked whether proper video is being fetched and rendered	High	Proper corresponding video is fetching and rendering	PASS
Fetch Reading material	2.4	Checked whether desired corresponding reading material is being fetched	High	Proper reading material is fetching	PASS
Fetch Quiz section	2.5	Checked whether desired quiz is being fetched	High	Proper quiz content is fetching	PASS
Show score	2.6	Check whether proper score is displayed according to user inputs	High	Correctly calculating and displaying the user score	PASS
Update the progress of user	2.7	Check whether the progress of the user is uploaded to the database properly	Medium	User progress is properly updated on the database.	PASS
View posts with real time updates	3.1	A page where an user can view posts made by other users which updates in real time.	High	Posts are loaded on frontend in real time	PASS
Like/Unlike posts	3.2	Users can like/unlike the posts of other users which updates in real time.	Low	Like on posts update real time	PASS
Add posts	3.3	Users can create and add	High	Posts are added in real	PASS

		their own posts that are added in real time to the feed.		time	
Add comments	3.4	Users can create and add comments on any post which updates in real time.	Medium	Comments are added real time	PASS
View comments	3.5		Medium	Comments are loaded on the frontend in real time	PASS
ASL to text	4.2	A page where an user can use their device camera to translate ASL to text.	High	The translation returns accurate predictions.	PASS

Regression testing

Under this strategy, we use the automated test suite (made using GitHub Actions). Whenever any pull request is made, the pulled code is subjected to this test automatically. This test sets up a build environment, fetches all the dependencies and then generates the final build. If at any point in this process the build fails, then the regression test stops and the developer is requested to review their code again.