How would you approach testing of this app?

a. Provide rough estimation and resources needed for testing this application.

- Rough estimation needed for testing this application: 3 days.

-Resources needed for testing this application:

Postman installed
Cypress installed
Chrome/Mozilla installed
Open Github account
Git installed
SelectorsHub installed

-Knowledge needed for properly testing this application:

Experienced in Cypress - Basics to Advanced + API requests.
Basic understanding of Xml and Json.
Good understanding of Git and GitHub basics.
Good understanding of Test Case & Bug Report Writing .
Good understanding of Black and White Box Test Techniques.
Basic understanding of the structure of HTML and formatting with CSS.
Good understanding of REST API Testing and Postman mechanics.

b. State critical functionalities.

Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations. The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs.

These functionalities are important for this app:

Can users successfully log in to the application once they provide legitimate credentials?

(Automated test cases for *log in* are in the "Cypress" section)

Can users successfully browse galleries and create galleries? (Bug reports for *browsing galleries* is in the "Bug reports" section.)

Can user successfully edit his galleries? (test cases for *editing galleries* are in "Test cases" section, also automated test cases for *editing galleries* are in "Cypress" section)

Do all the buttons work properly?

Is there any visual discomfort for the user?

Is App protecting us against explicit words and images?

Are there any legal copywriting issues?

c. State testing types you would suggest for this application.

Unit testing

Unit tests are very low level and close to the source of an application. They consist in testing individual methods and functions of the classes, components, or modules used by your software. Unit tests are generally quite cheap to automate and can run very quickly by a continuous integration server.

Integration testing

Integration tests verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database or making sure that microservices work together as expected. These types of tests are more expensive to run as they require multiple parts of the application to be up and running.

Functional testing

Functional tests focus on the business requirements of an application. They only verify the output of an action and do not check the intermediate states of the system when performing that action.

There is sometimes a confusion between integration tests and functional tests as they both require multiple components to interact with each other. The difference is that an integration test may simply verify that you can query the database while a functional test would expect to get a specific value from the database as defined by the product requirements.

End-to-end testing

End-to-end testing replicates a user behavior with the software in a complete application environment. It verifies that various user flows work as expected and can be as simple as loading a web page or logging in or much more complex scenarios verifying email notifications, online payments, etc...

End-to-end tests are very useful, but they're expensive to perform and can be hard to maintain when they're automated. It is recommended to have a few key end-to-end tests and rely more on lower level types of testing (unit and integration tests) to be able to quickly identify breaking changes.

Acceptance testing

Acceptance tests are formal tests that verify if a system satisfies business requirements. They require the entire application to be running while testing and focus on replicating user behaviors. But they can also go further and measure the performance of the system and reject changes if certain goals are not met.

Performance testing

Performance tests evaluate how a system performs under a particular workload. These tests help to measure the reliability, speed, scalability, and responsiveness of an application. For instance, a performance test can observe response times when executing a high number of requests, or determine how a system behaves with a significant amount of data. It can determine if an application meets performance requirements, locate bottlenecks, measure stability during peak traffic, and more.

Smoke testing

Smoke tests are basic tests that check the basic functionality of an application. They are meant to be quick to execute, and their goal is to give you the assurance that the major features of your system are working as expected.

Smoke tests can be useful right after a new build is made to decide whether or not you can run more expensive tests, or right after a deployment to make sure that they application is running properly in the newly deployed environment.

d. Describe which sequence/timeline of testing you would suggest,

coverage, scope...

Sequence: There are four main stages of testing that need to be completed before a program can be cleared for use: unit testing, integration testing, system testing, and acceptance testing.

Coverage: Bug reports, Manual test cases, Automated test cases and test cases via API backend (using Postman or Cypress) on "Register" section, "Login" section, "All Galleries" section, "Create Gallery" section, "My Galleries" section, "Edit Gallery" "Delete Gallery" and "Comment" sections. Also see if all the links for buttons are live and up and running. Check if CSS visuals are up to date and check if JS scripting is up to date for every part of the app.