1. As It is mentioned, That assigned app has some logical changes In the back-end so it's better to start with manual Testing

   Test Planning Process

   I use a combination of **Software Testing Methods: Agile, Black Box , white box ,Manual and Automated**

   A. Unit

   Manual Testing:

   I'd Start with manual Testing and Then Automation Testing
   I'd first Analyze and review the Requirements. I review the application to see how it Works and Identify appropriate inputs or outputs to determine How the app should be tested. And of course, It's Completely Based on the how the end user will consume the system hence this has to Be measurable, detailed and meaningful.

   The team Determines The scope of Testing(who is dedicated for FVT, Who is dedicated for SVT , etc)

   After that The team Designs a Test Startegy:
   I'd First assign People to do functional Testing to see how the product works.
   In case I'm familiar with main source code I do white box and In case I'm not, I do black box testing
   And Finally I Start Developing Manual Test Cases and Test Scenarios

   After creating Test Plan according to People, I start creating test suites considering all modules of application and then I write Test cases for every unit Using **QASE and Trello.**
   After Writing Test cases, I start reviewing Them(or have them reviewed by our Team Lead) and then I start executing them to detect bugs, If I caught one, I'd report them and once It's fixed I will re-run failed Test cases.

   After that. I Start Writing Test Scenarios(**Integration Testing**) for units of application for seeing the combination of their functionality altogether. And I Execute them.
   And We can Do **Smoke Testing** Immediately And Verify major and Critical Functionalities and We can Reject Build If We Found Important Bug
   The Unit Testing Level Is Finished
   B.
   Integration Testing

In Integration Testing I Start Combining My Test Cases and Test Scenarios For Monitoring Data Flow, For Java Stack We can Use **TestNG, Junit, Jmockito)** and for Python we can Use **Robot Framework**

And Now It's Time To write Test Scenarios Based On The different Sections As we want to combine different software modules and check the data flow.

C.

System Testing

And Now It's Time To Combine our Test cases and Scenarios and perform them in a complete, Integrated System, This allows chekinh system's compliance as per the requirements. and Now it's Time To do Performance and Load Testing, We can Use Tools like **Jmeter, LoadRunner Loadview**, and For Python, The Best Match is **Locust.**
Stress Testing can also be done Using Mentioned Tools

It can Be Done By Customers Too But Just in case, Testers Need To Do **Sanity testing** for the Stable Build version to verify Minor changes or bug fixes.
Testers can reject the Build if Sanity Test Fails.

D.

Acceptance Testing

And Finally We can do **monkey Testing** by giving random inputs to application new features to see the performance.
It's Time to Install Stable build on Real Devices to do UAT (User Acceptance Testing) but other Stackholders can be involved Too.
.

New Test Cases/Scenarios Depending on new Business Requirements Will be added

Automation Testing

What To Test?

I start testing **Database**(We can use **Data factory Tool** For example):

- ACID properties and CRUD operations
- Database Schemas
  (Primary Keys, Foreign keys, Field names , etc)
  Using Tools like **SchemaCrawler**
- Triggers:
  (white box: insert or update a new users' data. It's better to test DB alone before the integration with front-end is made)
  (blackbox: directly load the data or insert it the way that invokes the Trigger)
- Stored Procedures
- Field Constrains

- (Default Values: Perform a front-end operation which exercises the Database object condition)

And Finally

Checking Data Integrity

API Testing

We can Do it Manually using **Postman**. Based on our back-end stack. We can also use **SoapUI** or **SwaggerUI** (Dedicated for FASTAPI)

There are other Tools Specific for Testing Which Can Separately be Used for API Testing Like **Selenium, Cucumber or Robot Framework**

- API Installation:
  Checking the respective folders (e.g : in an android app)
  To make sure that the files/elements have made to their target
  Folders in the way they are supposed to.

- Return Value Based on Input Condition:
  Can be done Using Postman or be Automated

- Does not Return anything:
  When there is no return value. A Behavior of API should be checked.

- Trigger other APIs
  If an output of an API triggers some event or interrupt, then those events and interrupt listeners should be tracked

- Update data structure:

- Modify Certain resources :

  Updating data structure will have some outcome or effect on the system, and that should be authenticated

  If API call modifies some resources then it should be validated by accessing respective resources
  E

And Finally. We Can do Main Automation Testing Using Desired Tools

(Selenium, Appium, Cypress , ….)