Agile manual testing lab practice questions based on real-life scenarios:

Scenario: Your Agile team is developing a new feature for a mobile application that requires users to enter their personal information. How would you approach testing the input fields to ensure that the data is being validated and stored correctly?

Test for Valid Input: You can start by testing the input fields to ensure that the data entered is valid. For example, you could test that the user's email address has the correct format, the password meets the complexity requirements, and the phone number is in the correct format.

Test for Invalid Input: You can then test for invalid input by intentionally entering incorrect or malformed data. This would help ensure that the application properly handles and rejects invalid input.

Test for Edge Cases: Test for edge cases, such as entering a large number of characters into the input field or entering special characters. This would help identify any limitations or unexpected behavior of the input fields.

Test for Data Storage: You can also test that the data entered into the input fields is being stored correctly in the database. You can do this by verifying that the data entered is accurately displayed in the user's account and that the user can access and edit their personal information.

Test for Security: Lastly, you can test the security of the application by checking that user's personal information is stored securely and is not accessible to unauthorized users.

Scenario: Your Agile team is working on a project that requires the use of a chatbot. How would you test the chatbot to ensure that it is providing the correct responses to user queries?

Develop Test Cases: Create a set of test cases based on the expected user queries and the responses that the chatbot should provide. Ensure that the test cases cover a wide range of possible user inputs and expected responses.

Test the Chatbot's Responses: Manually test the chatbot by entering user queries and verifying that the chatbot responds with the correct answers. You can also test the chatbot's ability to handle unexpected inputs and provide appropriate responses.

Use Automated Testing Tools: Use automated testing tools to perform regression testing and ensure that the chatbot still functions correctly after making changes to the code.

Conduct User Acceptance Testing: Conduct user acceptance testing by having real users interact with the chatbot and provide feedback on its responses. This would help identify any potential areas for improvement and ensure that the chatbot is meeting user needs.

Monitor the Chatbot's Performance: Monitor the chatbot's performance to ensure that it is providing accurate responses in real-time. You can use metrics such as response time and user feedback to evaluate the chatbot's performance.

Scenario: Your Agile team is developing a new feature for a web application that requires the use of a file upload system. How would you test the file upload system to ensure that files are being uploaded correctly and that the correct file types are being accepted?

Test for Valid File Types: Start by testing the file upload system to ensure that it accepts only valid file types. For example, if the system is designed to accept only image files, you should verify that it rejects other file types such as PDFs or text files.

Test for File Size Limits: Test that the file upload system is capable of handling files of different sizes. This would help identify any limitations or unexpected behavior of the system.

Test for Invalid File Inputs: Test for invalid file inputs by intentionally uploading files that are corrupted, empty or with malformed file names. This would help ensure that the application properly handles and rejects invalid files.

Test for Security: Test the security of the file upload system by ensuring that uploaded files are not accessible to unauthorized users, and that the system does not allow execution of malicious scripts.

Test for Data Integrity: Test that uploaded files are stored correctly and that they can be accessed and downloaded by authorized users.

Test Performance: Test the performance of the file upload system to ensure that it is responsive and can handle multiple file uploads simultaneously.

Conduct User Acceptance Testing: Conduct user acceptance testing by having real users upload files and provide feedback on the system's performance.

Scenario: Your Agile team is working on a project that requires the use of a payment gateway that has specific requirements for payment information. How would you approach testing the payment information to ensure that it is being collected and processed correctly?

Test for Valid Payment Information: Test the payment gateway to ensure that it accepts valid payment information. For example, you should test that the system accepts valid credit card numbers, expiration dates, CVV codes, and other required information.

Test for Invalid Payment Information: Test for invalid payment information by intentionally entering incorrect or malformed data. This would help ensure that the application properly handles and rejects invalid payment information.

Test for Edge Cases: Test for edge cases, such as entering a large number of characters into the input field or entering special characters. This would help identify any limitations or unexpected behavior of the payment gateway.

Test for Payment Processing: Test that the payment gateway is processing payments correctly and that payments are being authorized and processed in a timely manner.

Test for Security: Test the security of the payment gateway to ensure that sensitive payment information is stored securely and is not accessible to unauthorized users.

Test Performance: Test the performance of the payment gateway to ensure that it can handle a high volume of payment transactions and is responsive to user requests.

Conduct User Acceptance Testing: Conduct user acceptance testing by having real users make payments and provide feedback on the payment process.

Scenario: Your Agile team is developing a new feature for a web application that requires the use of location data. How would you test the location data to ensure that it is being retrieved correctly and accurately?

Test for Valid Location Data: Test the location data to ensure that it is being retrieved correctly and that it matches the actual location of the user. For example, you should test that the system retrieves the correct latitude and longitude coordinates.

Test for Invalid Location Data: Test for invalid location data by intentionally entering incorrect or fake location data. This would help ensure that the application properly handles and rejects invalid location data.

Test for Different Locations: Test for different locations to ensure that the system can retrieve location data accurately from different regions and countries.

Test for Performance: Test the performance of the location data retrieval system to ensure that it is responsive and can handle a high volume of location requests.

Test for Security: Test the security of the location data retrieval system to ensure that it is not accessible to unauthorized users and that sensitive location data is stored securely.

Conduct User Acceptance Testing: Conduct user acceptance testing by having real users use the location feature and provide feedback on the accuracy and responsiveness of the location data.

Test Integration with Other Features: Test the integration of the location feature with other features of the web application, such as the map display or location-based search.

Scenario: Your Agile team is working on a project that requires the use of a database. How would you approach testing the database to ensure that data is being retrieved and stored correctly?

Test for Data Integrity: Test that data is being stored accurately in the database. For example, you should test that the data being stored matches the format and type of the data being entered.

Test for Data Retrieval: Test that data is being retrieved correctly from the database. This would include testing the accuracy and completeness of search queries, and verifying that data is being returned in the expected format.

Test for Performance: Test the performance of the database by performing load testing to ensure that it can handle a high volume of transactions and is responsive to user requests.

Test for Data Security: Test the security of the database to ensure that sensitive data is protected from unauthorized access, and that data is stored securely.

Test for Backup and Recovery: Test the backup and recovery process of the database to ensure that data can be restored in case of a failure or loss of data.

Conduct User Acceptance Testing: Conduct user acceptance testing by having real users test the database and provide feedback on its functionality, reliability, and ease of use.

Test Integration with Other Features: Test the integration of the database with other features of the application, such as user authentication, file uploads, or payment gateways.

Scenario: Your Agile team is developing a new feature for a mobile application that requires users to navigate through multiple screens. How would you test the navigation to ensure that users can easily move between screens and that the application does not crash?

Test Basic Navigation: Test the basic navigation of the application, including the use of buttons, icons, and menus. Ensure that users can navigate between screens with ease, and that all buttons and links are functioning correctly.

Test Navigation Flow: Test the flow of navigation by following a predefined user path and checking that the application follows the intended navigation flow. This would help ensure that the application is intuitive and easy to use for users.

Test Backward Navigation: Test the backward navigation of the application to ensure that users can easily move back to previous screens and that the application does not crash or freeze.

Test Navigation Across Different Devices: Test the navigation of the application across different devices and platforms, including iOS and Android, and ensure that the application provides a consistent user experience across all devices.

Test for Performance: Test the performance of the navigation by performing load testing to ensure that the application can handle a high volume of users and that the navigation is responsive and smooth.

Test Error Messages: Test error messages that may appear during navigation to ensure that they are clear, concise, and help users to navigate to the correct screen.

Conduct User Acceptance Testing: Conduct user acceptance testing by having real users test the navigation of the application and provide feedback on its ease of use and intuitiveness.

Scenario: Your Agile team is working on a project that requires the use of a search feature. How would you test the search feature to ensure that it is retrieving the correct results and that the results are being displayed correctly?

Test for Accuracy: Test the accuracy of the search feature by entering different search terms and checking that the results returned are relevant and accurate. Ensure that the search results include all relevant items and do not return any irrelevant items.

Test for Consistency: Test the consistency of the search feature by conducting multiple searches and checking that the results are consistent across different searches.

Test for Speed: Test the speed of the search feature by conducting multiple searches with a large volume of data and ensuring that the search results are displayed quickly.

Test for Error Messages: Test error messages that may appear during the search process, such as if the search term entered is invalid, and ensure that the error messages are clear and easy to understand.

Test for Pagination: Test the pagination feature that may be required if the search results are displayed on multiple pages, and ensure that the navigation through the search results is easy to use and accurate.

Test for Filtering: Test the filtering feature that may be required to refine search results and ensure that it is working correctly and that results are filtered accurately.

Conduct User Acceptance Testing: Conduct user acceptance testing by having real users test the search feature and provide feedback on its accuracy, speed, and ease of use.

Scenario: Your Agile team is developing a new feature for a web application that requires the use of forms. How would you test the forms to ensure that they are collecting the correct data and that the data is being validated correctly?

Test for Completeness: Test the completeness of the form by ensuring that all required fields are present and that they are marked as required. Check that the form includes all necessary fields to collect the required data and that none of the fields are missing.

Test for Valid Data: Test the form for valid data by inputting valid data in all the fields and ensuring that the data is being collected correctly. Verify that the data is being stored correctly in the database and that it is being displayed accurately.

Test for Invalid Data: Test the form for invalid data by inputting invalid data in all fields and ensuring that the form validation is working correctly. Check that the form is displaying error messages when invalid data is entered and that it is not accepting invalid data.

Test for Input Validation: Test the form input validation by entering data that exceeds the input field's character limit and ensuring that the form validation is working correctly.

Test for Required Fields: Test the required fields by submitting the form without entering data into any of the required fields, and ensure that the form validation is working correctly.

Test for Data Formatting: Test the form data formatting by inputting data that requires specific formatting, such as dates, phone numbers, and email addresses, and ensure that the form validation is working correctly.

Conduct User Acceptance Testing: Conduct user acceptance testing by having real users test the form and provide feedback on its usability, ease of use, and intuitiveness.