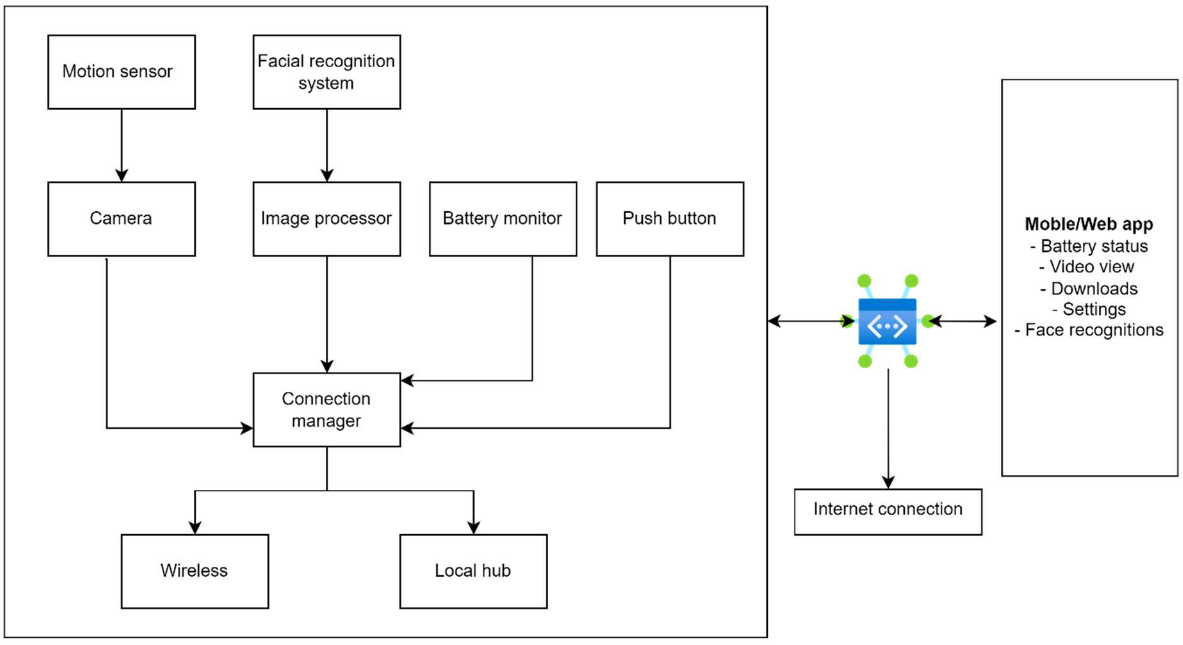
Task 1:

You are Test Lead for a new doorbell camera. The doorbell is battery powered, has one button, one camera, a motion sensor, and is wireless, which allows it to connect to a local hub, through which it can access the Internet. The camera settings and video are accessible through a mobile webapp, which also reports the battery life of the doorbell. The unique selling point of the product is its advanced facial recognition technology, which allows the user to identify known individuals on arrival. Videos are automatically tagged with known individual's names, all viewable via the webapp. During development, a simple serial connection is enabled via pins on the internal electronics of the product, which allow developers and testers to control and change settings directly on the doorbell module via a CLI. The CLI is disabled during production. The Test Manager would like you to focus testing efforts on automation and feature validation.

1.1 Create a simple system diagram of the system under test. Label and annotate your diagram for clarity.

System Diagram of Doorbell camera :

Below is the simple system diagram of the doorbell camera unit which in our case is system under test :



Components

Doorbell camera unit: This is the main h/w component. It includes the motion sensor, facial recognition system, camera, image processor, battery monitor, push button and a connection manager which further have Wireless and local hub connection implementation.

Local hub: Stands between the doorbell unit and web app

Web app: User interface for interacting with the doorbell system for various options like Battery status, configuration and monitoring, face recognitions, permissions etc.

1.2 Create a simple(top-level) test plan for the product. Include any assumptions made.

Top level test plan

1.2.1 Objective:

The purpose of test plan is to ensure that the doorbell camera functions including facial recognition, motion sensor, wireless connectivity, video capture, battery monitoring and integration with web app is as intended.

1.2.2 Scope

This project includes functional testing of all features, performance testing for the motion sensor and camera for facial recognition, battery life monitoring and UI testing for the web app with cross-platform support.

1.2.3 Assumption

1. The CLI/API is available for testing builds but is disabled in the production build.
2. The test environment simulates real-world conditions.
3. The mobile app is compatible with major operating systems (iOS and Android).

1.2.4 Features to test

1. **Motion sensor:** Verify that motion is correctly detected and triggers video recording.
2. **Facial Recognition:** Confirm that known individuals are correctly identified and tagged in the video.
3. **Video Recording:** Ensure good quality video capture and storage.
4. **Battery Checks:** Validate that battery life is accurately reported.
5. **User interface**: Ensure all functions, such as video play, configuration of parameters, and battery life display, works as expected.

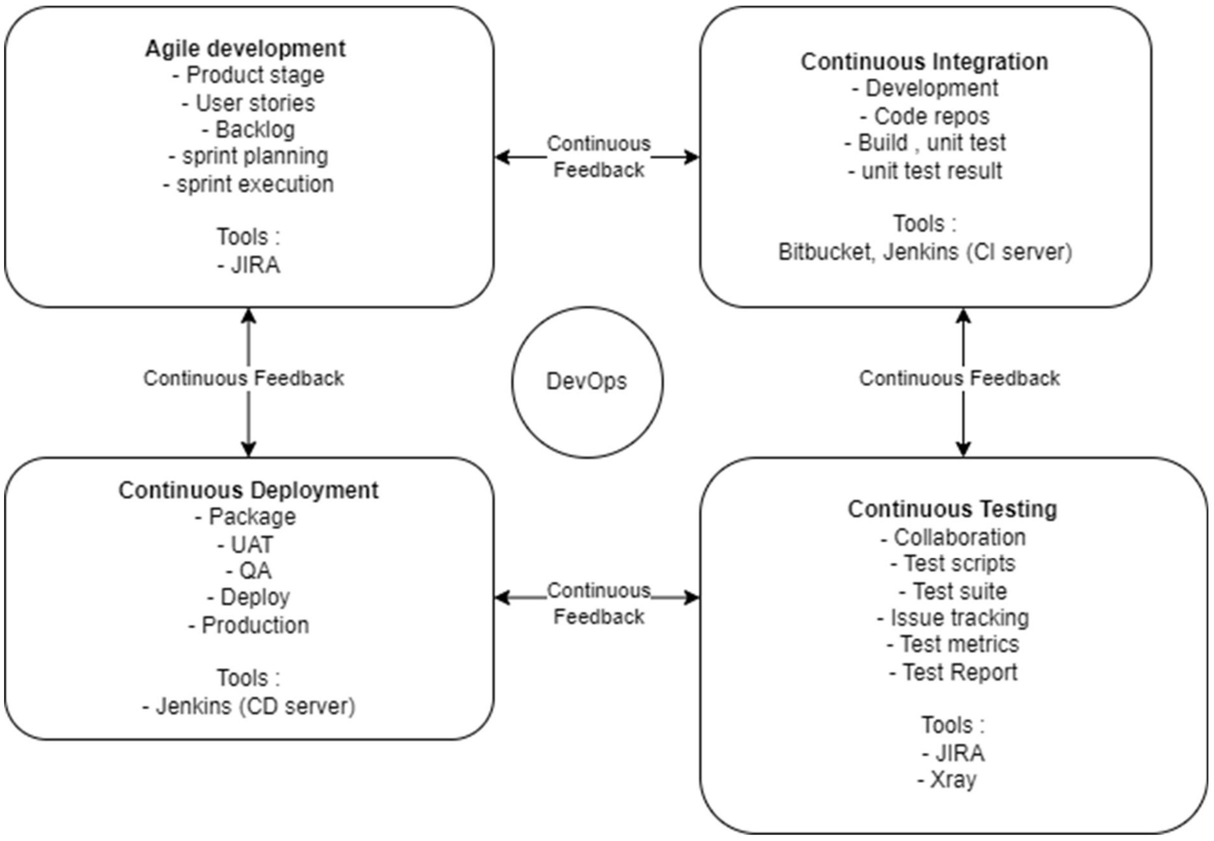
1.2.5 Test Environment

1. **Simulation:** Doorbell unit simulator/physical device.
2. **Devices:** Multiple mobile devices for app testing (both iOS and Android).

1.2.6 Test Phases

1. Unit Testing: Conducted by developers to ensure each module works correctly.
2. Integration Testing: Ensuring modules work together.
3. System Testing: end-to-end testing in a controlled environment.
4. User Acceptance Testing (UAT): Conducted by users to validate usability and functionality.

1.3 Propose a test strategy for the product. Include equipment and tools you might need. Highlight any area where developer support might assist in automation of features.



1. **Automation Scope:**
   a. Automate functional tests, including motion detection, video recording, battery monitoring, and facial recognition.
   b. Automate regression tests to ensure that new updates do not introduce new bugs.
   c. Automate performance tests to measure response times and battery consumption.
2. **Tools and Equipment:**
   a. Test Framework: Use a Python's PyTest for scripting automated tests and selenium for automating UI tests.
   b. CI/CD Pipeline: Integrate tests into a CI/CD pipeline using Jenkins or bitbucket. Run automated tests on every code commit or pull request.
3. **Developer Support:**
   a. Ensure developers provide clear API documentation for test automation scripts.
   b. Request developers to create mocks for components still developing or if they are difficult to test in live environment.
   c. Collaboration on creating a logging mechanism to capture detailed information during test execution.
4. **Automated Test Execution:**
   a. Schedule overnight test execution for full regression testing.
   b. Run critical tests after each code check-in.

c. Provide test reports and logs to developers for quick feedback and debugging. Create bug reports in JIRA for management tracking.

1.4 Give a detailed example of an automated test case. Your test should cover validation of a facial recognition function. Provide details of the tools and techniques used.

Example automation test

1. **Set Up:**
   a. Initialize the doorbell camera and ensure it is connected to the local hub.
   b. Pre-load the facial recognition system with expected images of known individuals using the CLI interface in the test environment.
   c. simulated test environment.
2. **Execution**:
   a. Simulate Motion: Trigger the motion sensor manually or using a device to simulate an approaching individual.
   b. Test Known Individuals: Use images or videos of individuals walking toward the camera.
   c. Capture Video: Allow the doorbell camera to record the video.
   d. Check tagging: Verify that the recorded video is correctly tagged with the known individual's name in the mobile web app.
   e. Test Unknown Individuals: Repeat the process using images of unknown individuals to ensure the system does not incorrectly tag videos.
3. **Assertions:**
   a. Check that the facial recognition system identifies know individuals correctly.
   b. Confirm that no incorrect tags are applied to videos of unknown individuals.
   c. Validate that all recordings are accessible and correctly labeled in the mobile app.
4. **Tools and Techniques:**
   a. Python Scripting: Write the test script using PyTest to automate the above steps.
   b. Selenium: Automate mobile app interactions to verify tagging and video playback.
   c. CLI Commands: Use the CLI to control during test setup and teardown.
   d. Image Recognition Tools: Use OpenCV or a similar library to automate verification of the camera's output against expected results.
5. **Test Results/Logs:**
   a. Log all actions taken by the script, including any discrepancies found.
   b. Generate a report summarizing test results, including pass/fail status and error logs for any failures.
6. **Post condition:**
   a. Reset the doorbell camera to its default state.
   b. Remove all test data from the facial recognition system.

Task 2:

Touch Surgery - Test Web Assignment

Instructions

This assignment will test your skills with test automation and your selected language

1. Using a test automation framework, create a script (as complex or simple as you wish) to:
    1. Open the website https://www.reddit.com/
    2. Search for a subreddit called "gaming"
    3. Open the sub-reddit
    4. Print out the top most post's title
    5. Perform a login
    6. Downvote the second post if it's upvoted already, upvote otherwise (in case the second post is an advertisement or announcement, use the third)
2. After you finish, *zip all your source code* and email us back

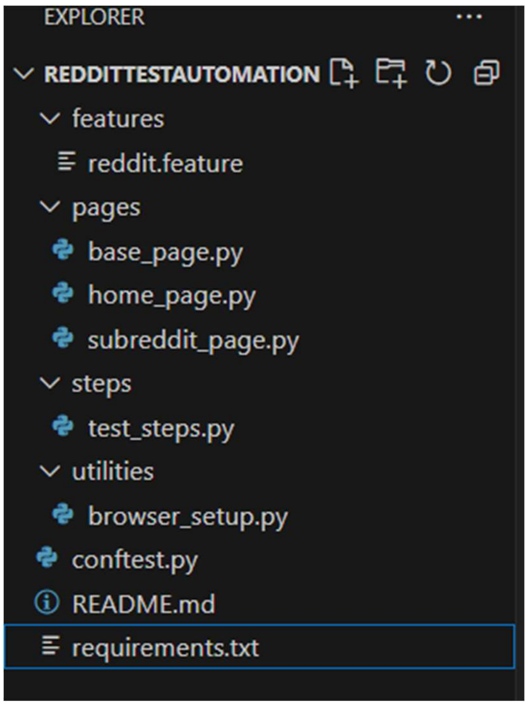
Extra points for:

1. Usage of BDD
2. Page-object pattern
3. Usage of good code standards
4. Documentation
5. Javascript/WebdriverIO (but remember, it's not necessary)

1. Overview :

a. To accomplish this task, create a test automation script in Python using Selenium WebDriver, which is a tool for automating functional tests in browsers. Provided is a structured solution following the Page model and using Behavior-Driven Development (BDD) with **pytest-bdd**. This approach helps to separate the logic of the test cases from the technical details of the web elements, promoting reusability and maintainability.

2. Project structure :



```
EXPLORER ...
∨ REDDITTESTAUTOMATION
  ∨ features
      reddit.feature
  ∨ pages
      base_page.py
      home_page.py
      subreddit_page.py
  ∨ steps
      test_steps.py
  ∨ utilities
      browser_setup.py
    conftest.py
    README.md
    requirements.txt
```

3. File/Class descripton :
    1. Features/reddit.feature: This file defines the BDD scenarios
    2. Pages/base_page.py: base class for all pages.
    3. Pages/home_page.py: Handles actions on home page
    4. Pages/subreddit_page.py: Handles actions on subreddit page.
    5. Steps/test_steps.py: Defines the steps to test all behaviours defined in feature file.
    6. Utilities/browser_setup.py: setup the browser for tests.
    7. Conftest.py : defines all reusable fixtures.
    8. Requirements.txt : requirements for the project.
    9. Readme.md : documentation on how to setup and run the tests.
4. Test execution :

   Follow readme file for details.

Task 3:

Touch Surgery - Android QA Assignment

Instructions

This assignment will test your skills with Espresso and your selected language (preferably Kotlin).

1. Clone this repository
2. Build the Android app under /TheSurgeonsTodoList to make sure it works fine on an emulator with API 27
3. Using Espresso, create tests (as complex or simple as you wish) that verify the following scenarios:
    1. Create new items

    2. Create new items with different priorities
    3. Delete a specific item
    4. Delete all items
    5. Change settings and check if they work
4. After you finish, *zip all your source code* and email us back *or* commit to a public github repository

Extra points for:

1. Robot pattern
2. Usage of good code standards
3. Documentation
4. Kotlin (but remember, it's not necessary)

1. Set up environment :

a. Git clone https://bitbucket.org/touchsurgery/android-test-assignment.git
b. Download android studio
c. Build and run app with API 27 (Android 8.1 oreo)
d. Used “Appium” to write tests in python.
    i. npm install -g Appium
e. Install Appium python client
    i. pip install Appium-Python-Client
f. Setup Appium server
    i. Appium
g. Configure Appium for android: need to make sure that emulator is running.

2. Writing auto tests :

a. test_todolist.py Class: This class sets up the driver and have all the auto tests in it.
b. The following test scenarios are implemented:
    i. Create New Items: Verify the ability to create new items in the to-do list.
    ii. Create New Items with Different Priorities: Validate that item can be created with different priorities.
    iii. Delete a Specific Item: Test the functionality of deleting a specific item from the list.
    iv. Delete All Items: Ensure that all items can be deleted from the list.
    v. Change Settings and Check if They Work: Verify changes in settings reflect correctly in the app.

3. Why appium and python :

a. Flexible language.
b. Cross platform
c. Extensive community support