Self-Introduction

I’m Rajat Kumar Pradhan, basically from Odisha and currently staying in Delhi. I completed my BCA from Berhampur University in 2018. I started my career at Test Well Technologies in Bangalore as a Junior Automation Test Engineer, where I worked for 2 years, gaining hands-on experience with Selenium WebDriver, Java, Postman, SQL, and Jenkins.

Currently, I am working at JIVEWS Private Limited, where I develop and execute test cases for regression and cross-browser testing, create automation scripts using the Page Object Model, and have been working with Playwright and JavaScript since last 3 months.

I have 3.6 years of experience and my technical skill includes Java, Selenium, Rest assured, Postman, Azure Pipeline, Bitbucket, SQL, Playwright, JavaScript and Cucumber Design pattern.

On my daily basis, I Review test plans to ensure alignment with project requirements.

Collaborate with my team to understand new functionalities and how to test them.

Write and update automated test scripts using Selenium and TestNG.

Execute automated test suites to detect failures.

Log and track defects using JIRA and work closely with developers for quick resolutions.

Participate in code reviews, providing and receiving feedback.

Integrate tests into the CI/CD pipeline for continuous testing.

Generate test reports and share with the team.

Attend meetings to provide status updates and discuss any issues encountered.

Dedicate time to learning and experimenting with new tools and techniques to improve my testing capabilities.

Project Explanation

Project Name: JIVIEWS – Workforce Management Solution

Overview: JIVIEWS is a system for the port industry that helps manage and deploy workers efficiently. It helps port teams in selecting workers, assigning tasks, deploying them, and ensuring accurate payment. It ensures fair work distribution among employees.

Role-Based Modules: The project has role-based modules like 'ADMIN' and 'Employee'. I worked on the 'ADMIN' module, which includes:

1. System Setup
2. System Definition
3. Employee Administration
4. Workforce Scheduling
5. Employee Self Service
6. Operation Planning & Execution
7. Time & Attendance
8. Reports
9. Dashboard

Features Focused On: Managing employees, scheduling work, tracking attendance, and handling operational planning.

I have automated modules like System Definition, Employee Administration, Workforce Scheduling, Time & Attendance, and Dashboard.

System Definition:

- Employee Setup: Configured employee details, including skills and roles.
- Roster Setup: Defined shifts and created rosters for role groups with rules.
- Schedule Creation Rules: Implemented scheduling and public holiday rules.

Employee Administration:

- Managed employee profiles, including IDs, addresses, and roles.

Workforce Scheduling:

- Automated scheduling of employees using Employee Roster and Workload Planner.
- Configured skill and role assignments for specific tasks like Crane Operators, QC, Clerks, and PM Operators.

Time & Attendance:

- Monitored employee attendance for accuracy.

Dashboard and Reports:

- Generated reports to analyze workforce performance and efficiency.

Responsibilities:

- Participated in all Agile processes.
- Developed and executed automation scripts using Selenium WebDriver, Java, and the Page Object Model (POM).
- Used Postman to test APIs.
- Documented and tracked defects in Jira.
- Conducted cross-browser testing on different browsers like Chrome and Firefox.

Q. In my current role, when I'm assigned a task

1. After writing the test script, I run the code on my local machine to ensure it's functioning as expected.
2. Once the local tests pass, I commit my changes to the repository and push the new code.
3. Our CI/CD pipeline, which is in Azure, automatically picks up the new code. I use a YAML file to define the build and test processes. The YAML file includes steps to run the tests, generate reports, and send an email to the team with the results.
4. The pipeline runs the integration tests, ensuring that the new changes don't break existing functionality, and then proceeds to deployment if all tests pass.

Q. How to do an automation task after assign a task to me.

1. I start by cloning the repository to my local machine using
   - **git clone <repository-url>**
   - cd <repository-directory>
2. I create a new branch for the task to keep my changes separate, using
   - **git checkout -b <new-branch-name>**
3. I regularly check the status of my working directory with **git status** to see what changes have been made.
4. I stage my changes using **git add <file-name>** to prepare them for commit.
5. I commit my changes with a meaningful message using
   - **git commit -m "Your commit message".**
6. I push my branch to the remote repository with
   - **git push origin <your-branch-name>**
7. Finally, I create a Pull Request for my changes to be reviewed and merged into the main branch.

Q. What is the difference between git pull and git fetch?

git pull fetches and merges changes from the remote repository to your local branch. git fetch only fetches changes without merging.

Q. How do you resolve merge conflicts in Git?

I resolve merge conflicts by manually editing the conflicting files, then adding and committing the resolved changes.

Q. Explain the use of branching in Git.

Branching allows multiple developers to work on different features without affecting the main codebase.

Q. How do you Revert a commit that has already been pushed and made public?

I use git revert to create a new commit that undoes the changes of the previous commit, ensuring the history remains intact.

Q. What is the difference between git reset and git revert?

git reset moves the branch pointer to a previous commit, effectively undoing commits. This can alter the commit history.

git revert creates a new commit that undoes changes from a previous commit, preserving the commit history.

Q. How do you delete a branch in Git?

To delete a local branch, use **git branch -d branch-name**.

To delete a remote branch, use **git push origin --delete branch-name**.

Q. What is a Git fork and how do you use it?

A Git fork is a personal copy of someone else's repository. It allows us to freely make changes without affecting the original project.

We can create a fork, make changes, and submit a pull request to propose our changes to the original project.

Q. Deploying builds to production issues

1. Dependency Conflicts:

- Sometimes, different versions of dependencies between development and production environments cause failures.

2. Configuration Issues:

- Incorrect configuration settings, such as database URLs or API keys, can cause unexpected behaviour.

3. Resource limitations:

- Insufficient memory, CPU, or storage can lead to performance decline or service outages, especially under high load.

4. Integration Failures:

- Issues can arise from integration points, such as third-party services or internal APIs, especially if they're not properly mocked or tested in pre-production environments.

5. Network Issues:

- Network connectivity problems can disrupt service availability or performance.

6. Database Issues:

- Schema mismatches, data migrations can cause failures or data corruption.

7. Logging and Monitoring Gaps:

- Lack of proper logging and monitoring can make it difficult to identify and troubleshoot issues quickly.

Q. How do you enhance user interface?

1. **Simplify Navigation**: Make sure users can easily find what they're looking for. Use clear menus, search functionality.
2. **Responsive Design**: Ensure our interface works well on all devices— desktops, tablets, and mobiles.
3. **Consistent Design**: Use a consistent style for fonts, colours, and buttons.
4. **Visual Hierarchy**: Organize elements so that the most important information stands out. Use size, colour, and positioning to guide users' attention.
5. **Feedback and Interaction**: Provide immediate feedback for user actions, like button clicks or form submissions.

6. **Accessibility:** Ensure our interface is usable by people with various disabilities
7. **User Testing:** Regularly test our interface with real users to identify pain points and areas for improvement.

Q. How do you define and implement a test plan?

A test plan defines the strategy, resources, scope, and schedule of the intended testing activities.
To create a test plan:

- **Identify the scope:** Understand the features to be tested.
- **Set objectives:** Define what you aim to achieve with the tests.
- **Determine resources:** Allocate the testing tools, personnel, and hardware.
- **Create a schedule:** Establish the timeline for testing.
- **Design test cases:** Based on the product features, specifications, and requirements.
- **Determine exit criteria:** Define when the test will be considered complete.

Q. How do you collaborate with cross-functional teams to derive test cases?

Collaboration involves regular meetings with product managers and developers to gather feature details, clarify requirements, and discuss functionality.
This allows QA engineers to:

- Understand the user stories or feature specifications.
- Ask questions to ensure there is no ambiguity.
- Derive test cases that cover positive, negative, edge, and boundary conditions.
- Prioritize testing based on the feature's business impact.

Q. How do you handle debugging and reporting issues?

- Use logs and API responses to trace the root cause of the issue.
- Reproduce the issue if possible and gather all the necessary details (e.g., steps, environment).
- Analyse failed test cases by reviewing the test data, environment, and code changes.
- Report bugs in a tracking tool Jira with detailed information, including steps to reproduce, screenshots, expected vs actual behaviour, and logs.

Q. What approach do you take to automate UI and API testing?

For UI automation:

- Use tools like Selenium to simulate user actions.
- Write maintainable and reusable test scripts.
- Incorporate frameworks like TestNG for test execution and reporting.

For API automation:

- Use tools like Postman or RestAssured.
- Automate API tests to validate request and response formats, status codes, and business logic.
- Integrate these into a CI/CD pipeline for frequent execution.

Q. How do you enhance an automation framework?

- Continuously improve the framework by adding new features or addressing existing issues.
- Introduce data-driven testing to make the scripts reusable.
- Improve the efficiency by optimizing wait strategies.
- Ensure the framework supports CI/CD integration for seamless automation.

Q. What is your experience with Agile methodologies and weekly sprints?

- I have experience working in Agile teams with weekly sprints.
- I participate in sprint planning to understand the user stories.
- Break down tasks, create test cases, and execute them within the sprint timeline.
- Participate in daily stand-ups, providing updates on testing progress and challenges.

Q. How do you handle testing for microservices and analyze API responses?

- I use tools like Postman to send API requests and validate the responses.
- Check status codes, response time, and the structure of JSON or XML data.
- Analyse micro services by checking logs and ensuring that services communicate properly, and handle faults gracefully.