Part 1. Test Plan Document: -

1. Introduction: -

2. Purpose of the test plan:

The purpose of this test plan is to outline the approach, scope, and methodologies for testing the web application. The goal is to ensure that the application is reliable, performs well, and meets the requirements specified in the project documentation.

3. Scope of testing:

Functionality related to logging into the application.
Validating the login functionality, editing, deleting, viewing locked messages.
Assessing edge cases and handling of errors related to locked messages.

- Page URL: - https://app.thepric.com/priclogin
- Targeted Browsers: Chrome, Firefox, Safari, Edge
- Targeted Devices: Laptop(OS-Windows 11)
- Test Environment: Development, Staging, Production, Testing,UAT
- Testing Types: Functional Testing, Usability Testing, Compatibility Testing, Performance Testing, Security Testing, etc.

4. Key Features:

- **User Authentication:** Secure login and user management.
- **Dashboard:** Overview of key metrics and data.
- **Data Management:** Input and manage [specific data].
- **Reporting:** Generate and view reports.
- **Notifications:** Alerts for updates and important events.

5. Test Approaches:

1. Manual Testing
2. Automated Testing (where applicable)
3. Cross-browser Testing
4. Accessibility Testing
5. Load Testing (if applicable)
6. Security Testing (including penetration testing)

6. Overview of the application under test:

Its web application, where users can manage messages.
It's a user-friendly interface where creators can easily create and share posts with hashtags, @mentions etc.
It includes features for logging in and interacting with locked messages and Broadcast, Telegram Subscription Manager.

2. Test Strategy:

2.1 Types of Testing to be Performed

1. **Functional Testing**
    - **Objective:** Ensure that each function of the application operates in conformance with the requirement specifications.
    - **Scope:** Test cases will cover all user interactions, workflows, and business logic.
2. **Security Testing**
    - **Objective:** Identify vulnerabilities and ensure data protection and secure access controls.
    - **Scope:** Includes penetration testing, vulnerability scanning, and assessment of authentication mechanisms and data encryption.
3. **Usability Testing**
    - **Objective:** Evaluate the user interface and overall user experience to ensure the application is intuitive and user-friendly.
    - **Scope:** Focuses on ease of navigation, accessibility, and overall user satisfaction.
4. **Compatibility Testing**
    - **Objective:** Verify that the application works across different browsers, devices, and operating systems.
    - **Scope:** Includes cross-browser testing, mobile responsiveness testing, and compatibility with various OS versions.
5. **Integration Testing**
    - **Objective:** Test interactions between integrated components or systems to ensure they work together as expected.
    - **Scope:** Includes testing of API integrations, third-party services, and data exchange between modules.
    -

2.2 Levels of Testing

1. **Unit Testing**
    - **Objective:** Test individual components or units of code for correctness.
    - **Scope:** Focuses on verifying the functionality of small, isolated parts of the application, such as functions or methods.
2. **Integration Testing**
    - **Objective:** Test the interactions and interfaces between integrated components or systems.
    - **Scope:** Ensures that different parts of the application work together and external integrations function correctly.
3. **System Testing**
    - **Objective:** Validate the complete and integrated application against the specified requirements.
    - **Scope:** Includes end-to-end testing of the entire application to ensure that it behaves as expected in a fully integrated environment.

3. Test Criteria:

3.1 Entry and Exit Criteria for Each Test Phase

2. Integration Testing

- **Entry Criteria:**
  - Ft testing of individual components is complete and defects have been resolved.
  - Integrated modules are available for testing.
  - Integration test cases are designed and reviewed.
  - Test environment is configured to support integration testing.
- **Exit Criteria:**
  - All integration test cases have been executed.
  - All critical and major integration issues have been resolved.
  - Interfaces between modules and external systems are functioning as expected.
  - Integration test results meet the defined success criteria.

3. System Testing

- **Entry Criteria:**
  - Integration testing is complete and all integration issues have been addressed.
  - System test cases are prepared and reviewed.
  - The complete application is deployed in a test environment that mimics the production environment.
- **Exit Criteria:**
  - All system test cases have been executed.
  - All critical and major defects identified have been resolved.
  - Application performance, security, and usability criteria are met.
  - System test results are documented and approved.

3.2 Pass/Fail Criteria for Test Cases

- **Pass Criteria:**
  - The actual result matches the expected result as defined in the test case.
  - All steps in the test case are executed without errors.
  - No critical or major defects are found during the execution of the test case.
  - The application behaves as specified in the requirements and design documents.
- **Fail Criteria:**
  - The actual result does not match the expected result.
  - Errors or issues occur during test execution that prevent the test case from completing.
  - Critical or major defects are identified that prevent the functionality from working as intended.
  - The application does not meet the requirements or design specifications.
  - Platform failure

4. Test Deliverables

4.1 Documents

1. **Test Plan Document**
   - **Description:** Detailed plan outlining the overall testing strategy, scope, objectives, and resources.
   - **Content:** Test strategy, types of testing, test phases, roles and responsibilities, and schedule.
2. **Test Case Specifications**
   - **Description:** Document detailing individual test cases, including objectives, inputs, execution steps, and expected results.
   - **Content:** Test case ID, description, preconditions, test steps, expected results, and postconditions.
3. **Test Data**
   - **Description:** Data sets used for executing test cases.
   - **Content:** Sample data for functional tests, data for edge cases, and any specific test scenarios.

4. **Test Environment Setup Document**
   - **Description:** Configuration and setup instructions for the test environment.
   - **Content:** Hardware and software requirements, environment configuration details, and setup procedures.

4.2 Reports

1. **Test Execution Report**
   - **Description:** Summary of the test execution results.
   - **Content:** List of executed test cases, pass/fail status, defects logged, and overall test coverage.
2. **Defect Report**
   - **Description:** Detailed report on defects identified during testing.
   - **Content:** Defect ID, description, severity, status, steps to reproduce, and resolution status.
3. **Test Summary Report**
   - **Description:** Comprehensive summary of the testing process and outcomes.
   - **Content:** Overview of testing activities, summary of results, defect trends, and overall assessment of application quality.
   -

4.3 Scripts

1. **Automated Test Scripts**
   - **Description:** Scripts used for automated testing.
   - **Content:** Test automation code/scripts for functional, regression, and performance testing. Includes test scripts for tools like Selenium, JMeter, etc.

2. **Test Execution Scripts**
    - **Description:** Scripts or procedures for executing test cases manually or automatically.
    - **Content:** Step-by-step instructions for running test cases, including setup and teardown procedures.

5. **Schedule**:

Test Plan Document Completion: 13-9-2024

Bug Reporting: 14-9-2024

Test Automation Script Development: 14-9-2024

Automation Execution: 15-9-2024

Final Test Report Submission: 16-9-2024

6. Risks and Mitigations

6.1 Potential Risks and Mitigation Strategies

1. Unclear Requirements

- **Risk:** Ambiguous or incomplete requirements may lead to inadequate test coverage or misaligned test cases.
- **Mitigation Strategies:**
    - **Requirements Review:** Engage with stakeholders to clarify and document requirements thoroughly.
    - **Frequent Communication:** Schedule regular meetings with stakeholders to address any ambiguities or changes in requirements.
    - **Requirements Traceability:** Use traceability matrices to ensure all requirements are covered by test cases.

2. Inadequate Test Coverage

- **Risk:** Missing test cases may lead to untested functionalities or scenarios, resulting in undetected defects.
- **Mitigation Strategies:**
    - **Comprehensive Test Planning:** Develop a detailed test plan and ensure all features and functionalities are covered.
    - **Test Case Reviews:** Conduct reviews and walkthroughs of test cases to validate coverage.
    - **Use Coverage Tools:** Utilize code coverage tools to measure and ensure adequate test coverage.

3. Resource Constraints

- **Risk:** Limited availability of testers, tools, or environments may delay testing activities.
- **Mitigation Strategies:**
    - **Resource Planning:** Allocate resources based on project needs and ensure they are available as scheduled.
    - **Cross-Training:** Train team members to handle multiple roles or tasks to cover for absences or shortages.
    - **Tool Selection:** Choose cost-effective or open-source tools if budget constraints are an issue.

4. Defect Management Issues

- **Risk:** High volume of defects or delayed resolution may impact the testing schedule and application quality.
- **Mitigation Strategies:**
    - **Defect Tracking System:** Implement a robust defect tracking system to log, prioritize, and manage defects effectively.
    - **Regular Defect Reviews:** Conduct regular reviews to assess defect status and ensure timely resolution.
    - **Clear Communication:** Maintain clear communication channels between testers and developers to expedite defect resolution.

5. Environment Issues

- **Risk:** Problems with the test environment (e.g., configuration issues, downtime) may hinder testing activities.
- **Mitigation Strategies:**
    - **Environment Setup:** Ensure that the test environment mirrors the production environment as closely as possible.
    - **Pre-Test Environment Validation:** Validate the environment setup before starting the testing phase.
    - **Backup and Contingency Plans:** Maintain backups and have contingency plans in place to address environment issues promptly.

6. Schedule Delays

- **Risk:** Testing activities may fall behind schedule, impacting project timelines and release dates.
- **Mitigation Strategies:**
    - **Detailed Scheduling:** Develop a realistic and detailed testing schedule with built-in buffer times for unexpected delays.
    - **Regular Progress Monitoring:** Track progress against the schedule and adjust resources or plans as needed.
    - **Prioritization:** Prioritize critical test cases and focus on high-impact areas to ensure essential functionalities are tested first.

7. Integration Issues

- **Risk:** Problems with integration between modules or third-party systems may affect testing outcomes.
- **Mitigation Strategies:**
    - **Integration Testing Early:** Start integration testing early to identify and resolve issues before they impact later phases.
    - **Mock Services:** Use mock services or simulators for third-party integrations if they are not available during testing.
    - **Close Coordination:** Work closely with development teams to address integration issues promptly.

8. User Acceptance Testing (UAT) Challenges

- **Risk:** End-users may encounter issues or have difficulties with the application, affecting UAT outcomes.
- **Mitigation Strategies:**
    - **Clear UAT Criteria:** Define clear UAT criteria and provide end-users with adequate training and documentation.
    - **Feedback Mechanism:** Implement a feedback mechanism to capture and address issues raised during UAT.
    - **Support:** Offer support to end-users throughout the UAT process to resolve issues quickly.