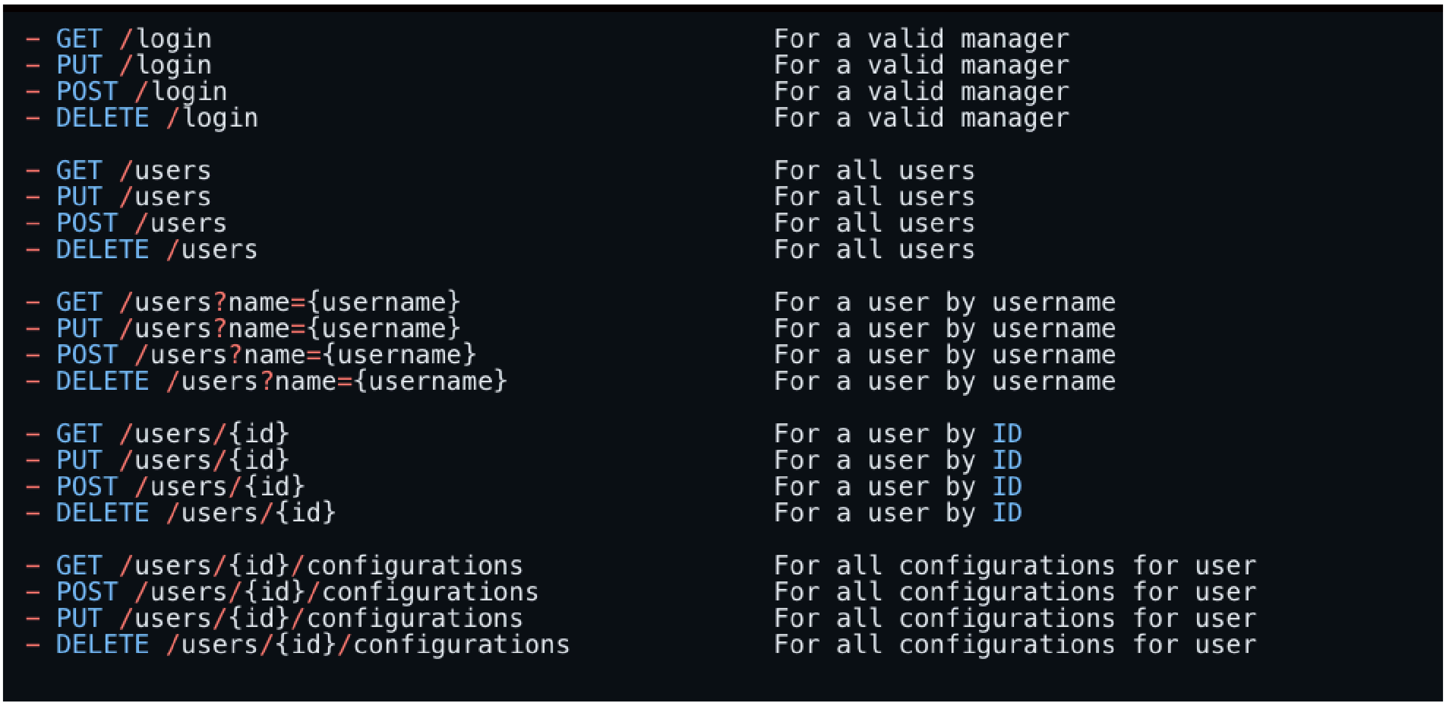
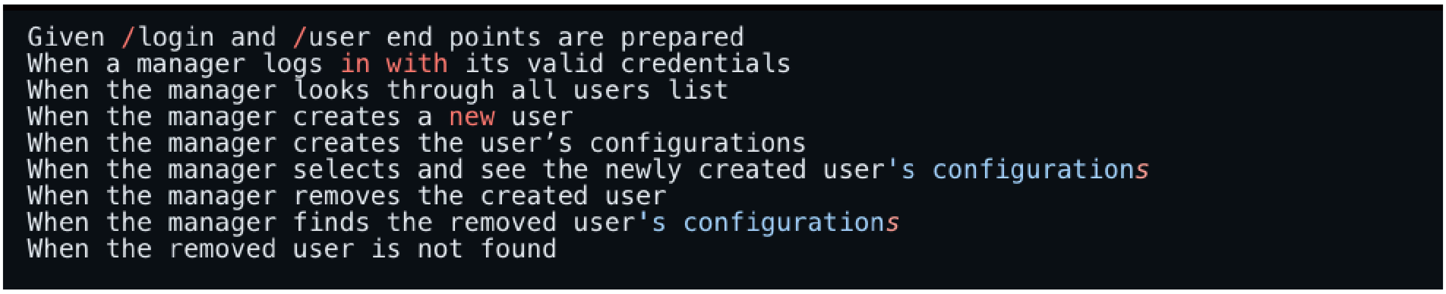
API Test Scenario

Restful API Endpoints



```
- GET /login                                    For a valid manager
- PUT /login                                    For a valid manager
- POST /login                                   For a valid manager
- DELETE /login                                 For a valid manager

- GET /users                                    For all users
- PUT /users                                    For all users
- POST /users                                   For all users
- DELETE /users                                 For all users

- GET /users?name={username}                    For a user by username
- PUT /users?name={username}                    For a user by username
- POST /users?name={username}                   For a user by username
- DELETE /users?name={username}                 For a user by username

- GET /users/{id}                               For a user by ID
- PUT /users/{id}                               For a user by ID
- POST /users/{id}                              For a user by ID
- DELETE /users/{id}                            For a user by ID

- GET /users/{id}/configurations                For all configurations for user
- POST /users/{id}/configurations               For all configurations for user
- PUT /users/{id}/configurations                For all configurations for user
- DELETE /users/{id}/configurations             For all configurations for user
```

Scenarios



```
Given /login and /user end points are prepared
When a manager logs in with its valid credentials
When the manager looks through all users list
When the manager creates a new user
When the manager creates the user’s configurations
When the manager selects and see the newly created user's configurations
When the manager removes the created user
When the manager finds the removed user's configurations
When the removed user is not found
```

Test Plan for Production Deployment

The test plan ensures that the web application is functioning correctly, is stable, and meets requirements in a production environment. It includes the following steps:

Objectives

- Validate API functionality
- Ensure data integrity and consistency across endpoints
- Validate E2E workflows
- Ensure the system can handle expected and peak load conditions
- Identify and mitigate security vulnerabilities

- Automate tests for efficiency and consistency via CI/CD pipelines

Types of Testing

Functional Testing:

- Focus on CRUD operations for APIs (/login, /users, /users/{id}/configurations)
- Validate API functionality
- Ensure data integrity and proper error handling

Load Testing:

- Ensure the application can handle concurrent users and requests
- Tool: K6 or Jmeter
- Simulate concurrent users to API endpoints

Stress Testing:

- Push the system beyond its limits to identify failure thresholds

Security Testing:

- **Unauthorized Access**: Attempt to access API endpoints without a valid token
- **Data Exposure**: Ensure sensitive data is encrypted
- **SQL Injection**: Attempt malicious queries on endpoints

Regression Testing:

- Verify that bug fixes or new features haven't broken existing functionality.
- Automated suite for functional, integration, and API tests

End-to-End Testing:

- Validate E2E API flows

Performance Testing:

- Analyze API response times for different operations under normal and peak conditions
- Ensure average response time is according to the defined performance benchmarks

CI/CD Pipeline:

- Setup automation tests to be triggered as part of CI/CD pipeline
- Tools: Jenkins, CircleCI or Github Actions

Testing Environments

- **Local Development Environment**: Initial testing and debugging
- **QA/Staging Environment**: Comprehensive testing that mirrors the production setup
- **Production Environment (limited access)**: Post Deployment validations

Test Requirements

- Access to the API endpoints
- Test environment mimicking production environment
- Test data for valid/invalid users and configurations

Test Data Example

- **Login Payload**: { "username": "manager_username", "password": "manager_password" }
- **Create User Payload**: { "name": "John Doe", "email": "johndoe@test.com" }
- **Add Configurations Payload**: { "configKey": "example_key", "configValue": "example_value" }

Test Plan Phases

Phase 1: Review the product Requirement Document (PRD) and gather requirements

Phase 2: Test Case Design, Preparation and Execution

- **Functional Tests**: Create detailed test cases covering all user interactions and workflows
- **Integration Tests**: Ensure that all APIs communicate correctly
- **Performance Tests**: Define load conditions (e.g., 1000 concurrent users)
- **Security Tests**: Write scripts for common vulnerabilities like SQL Injection

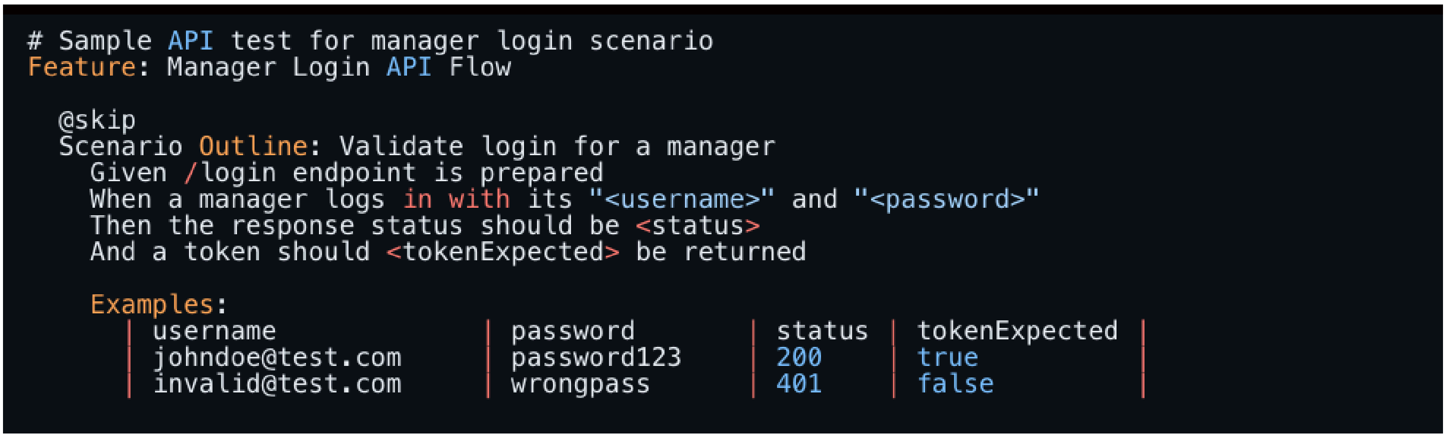
Example Test Case Design:

TEST CASE ID	DESCRIPTION	STEPS	EXPECTED OUTCOME
TC001	MANAGER LOGIN	1. Prepare /login Endpoint 2. Send /login request with valid credentails	1. The Endpoint returns 200 response 2. Token is returned in the response

Phase 3: Implement Automation

- **Cypress**: Implement POM structure for maintainable test scripts for backend and frontend application
- **K6**: Write performance scripts with scenarios for peak load testing.

Example Automation Test:



```
# Sample API test for manager login scenario
Feature: Manager Login API Flow

  @skip
  Scenario Outline: Validate login for a manager
    Given /login endpoint is prepared
    When a manager logs in with its "<username>" and "<password>"
    Then the response status should be <status>
    And a token should <tokenExpected> be returned

    Examples:
      | username             | password       | status | tokenExpected |
      | johndoe@test.com     | password123    | 200    | true          |
      | invalid@test.com     | wrongpass      | 401    | false         |
```

Phase 4: Execute Tests in CI/CD Pipeline

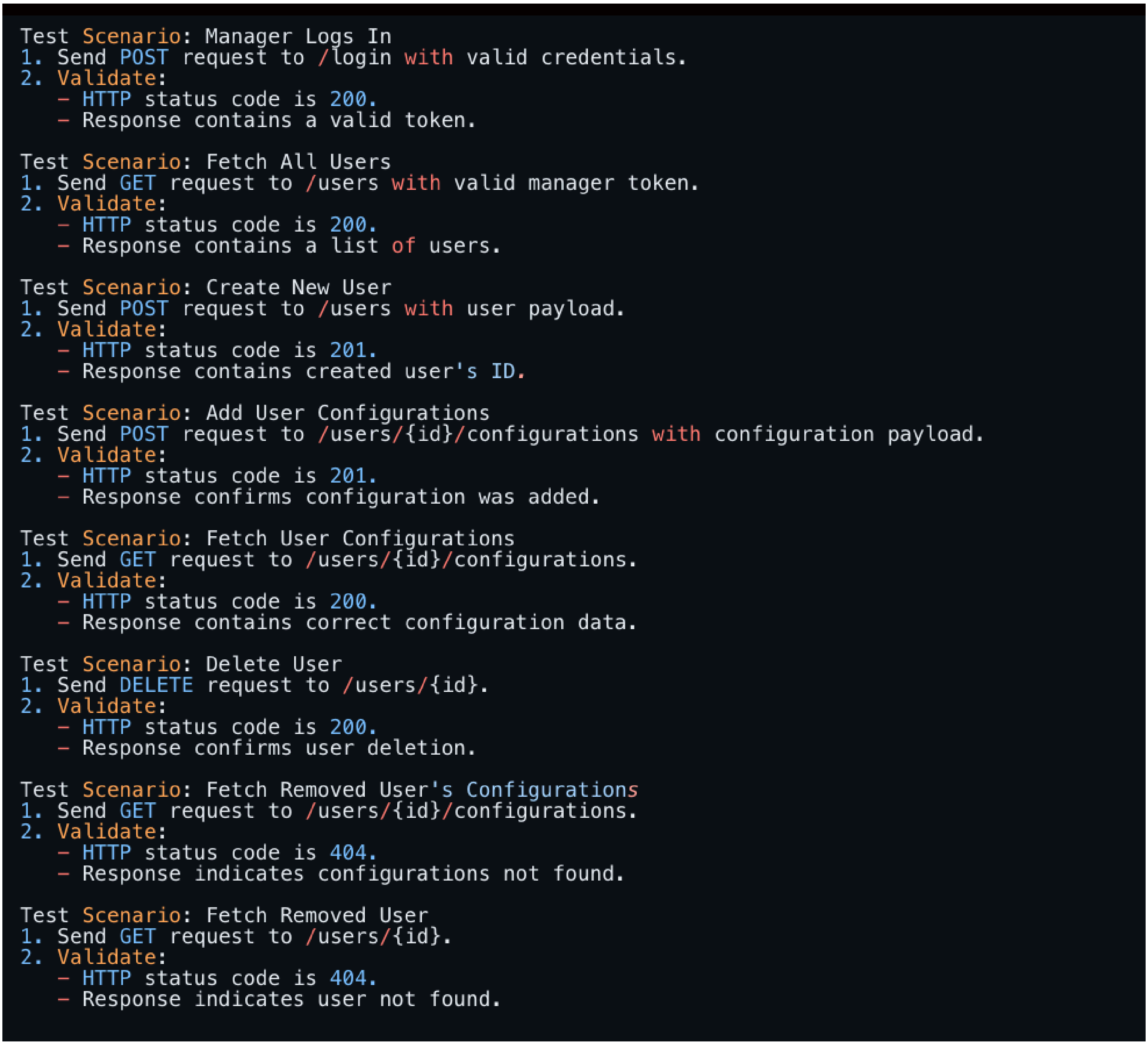
CI/CD Workflow:

- **Build**: Run unit tests and static code analysis.
- **Test Stage**:
  - **Step 1**: Run Cypress Functional, E2E and Integration tests.
  - **Step 2**: Execute performance tests with K6 and Security Tests.
- **Deploy to Staging**: Only if all tests pass.
- **Review**: Conduct Regression and UAT in the staging environment.
- **Deploy to Production**: Triggered once the UAT is approved.

Phase 5: Monitoring and Reporting

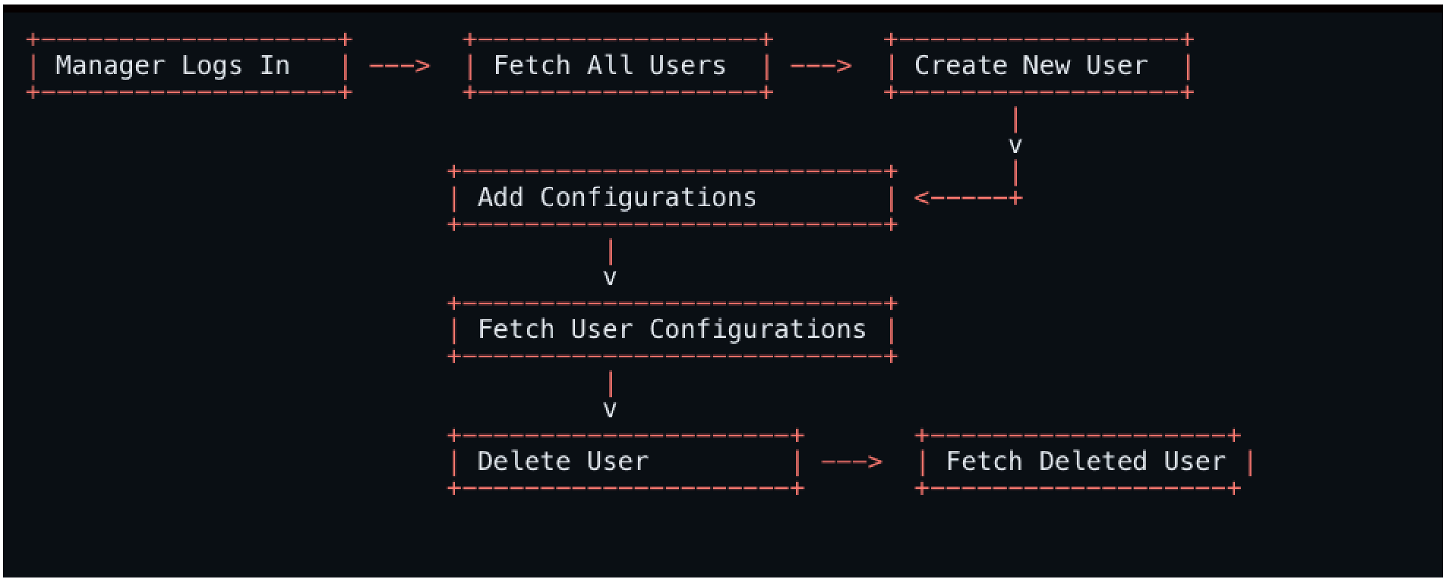
- Use monitoring tools to monitor application health post-deployment.
- Set up alerts for critical errors and performance degradation.
- Generate detailed test reports with Cypress and integrate them into the CI/CD dashboard.

Puesdo Code for API Sceanrios



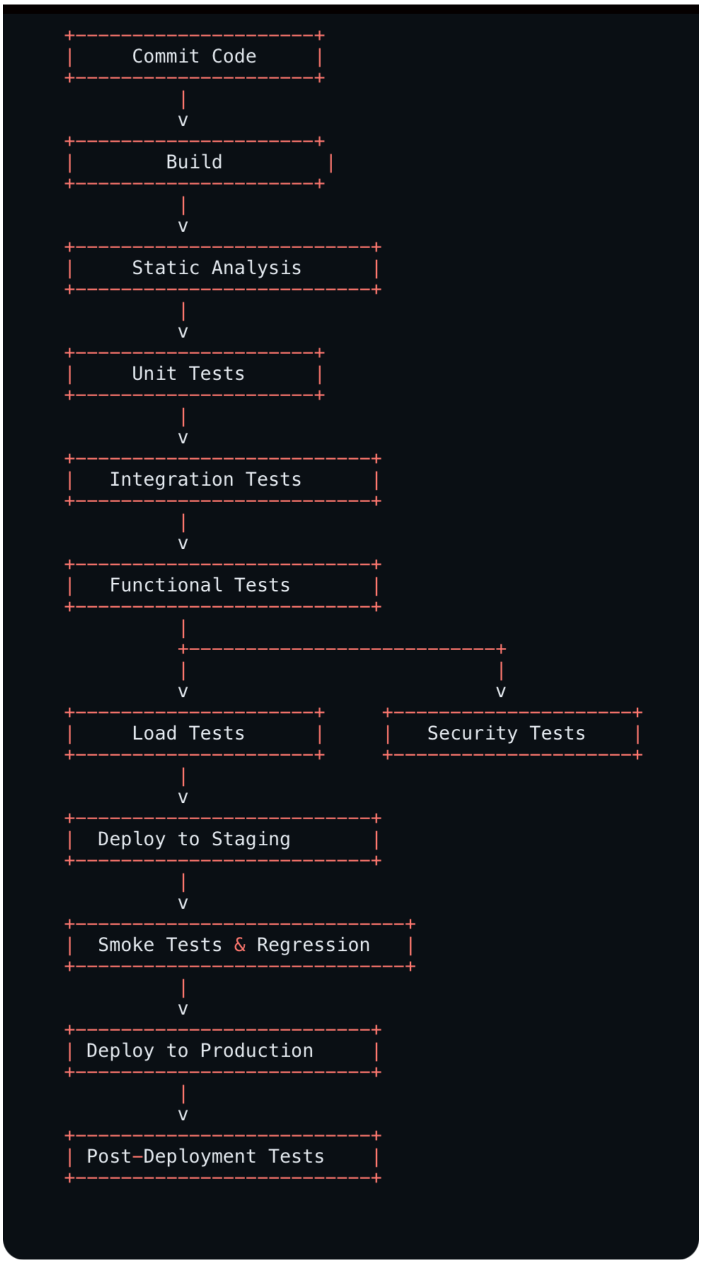
```
Test Scenario: Manager Logs In
1. Send POST request to /login with valid credentials.
2. Validate:
   - HTTP status code is 200.
   - Response contains a valid token.

Test Scenario: Fetch All Users
1. Send GET request to /users with valid manager token.
2. Validate:
   - HTTP status code is 200.
   - Response contains a list of users.

Test Scenario: Create New User
1. Send POST request to /users with user payload.
2. Validate:
   - HTTP status code is 201.
   - Response contains created user's ID.

Test Scenario: Add User Configurations
1. Send POST request to /users/{id}/configurations with configuration payload.
2. Validate:
   - HTTP status code is 201.
   - Response confirms configuration was added.

Test Scenario: Fetch User Configurations
1. Send GET request to /users/{id}/configurations.
2. Validate:
   - HTTP status code is 200.
   - Response contains correct configuration data.

Test Scenario: Delete User
1. Send DELETE request to /users/{id}.
2. Validate:
   - HTTP status code is 200.
   - Response confirms user deletion.

Test Scenario: Fetch Removed User's Configurations
1. Send GET request to /users/{id}/configurations.
2. Validate:
   - HTTP status code is 404.
   - Response indicates configurations not found.

Test Scenario: Fetch Removed User
1. Send GET request to /users/{id}.
2. Validate:
   - HTTP status code is 404.
   - Response indicates user not found.
```

API Workflow Diagram



```
+---------------------+        +-------------------+        +-------------------+
| Manager Logs In     |  --->  | Fetch All Users   |  --->  | Create New User   |
+---------------------+        +-------------------+        +-------------------+
                                                                       |
                                                                       v
                             +-----------------------------+           |
                             | Add Configurations          |  <--------+
                             +-----------------------------+
                                         |
                                         v
                             +-----------------------------+
                             | Fetch User Configurations   |
                             +-----------------------------+
                                         |
                                         v
                             +-----------------------+        +--------------------+
                             | Delete User           |  --->  | Fetch Deleted User |
                             +-----------------------+        +--------------------+
```

Deployment Workflow Diagram



```
+----------------------+
|     Commit Code      |
+----------------------+
          |
          v
+----------------------+
|        Build          |
+----------------------+
          |
          v
+----------------------------+
|     Static Analysis        |
+----------------------------+
          |
          v
+----------------------+
|     Unit Tests       |
+----------------------+
          |
          v
+----------------------------+
|   Integration Tests        |
+----------------------------+
          |
          v
+----------------------------+
|   Functional Tests         |
+----------------------------+
          |
          +----------------------------+
          |                            |
          v                            v
+----------------------+     +----------------------+
|     Load Tests       |     |   Security Tests     |
+----------------------+     +----------------------+
          |
          v
+----------------------------+
|  Deploy to Staging         |
+----------------------------+
          |
          v
+-------------------------------+
|  Smoke Tests & Regression     |
+-------------------------------+
          |
          v
+----------------------------+
| Deploy to Production       |
+----------------------------+
          |
          v
+----------------------------+
| Post-Deployment Tests      |
+----------------------------+
```

Risk Assessment

Potential Risks:

- Server load issues during peak traffic.
- Security vulnerabilities missed in scans.

Mitigation:

- Run stress and soak tests.
- Use real user monitoring tools.
- Regular security audits.

Entry and Exit Criteria

Entry Criteria:

- All unit and integration tests pass.
- Build completes successfully without errors.

Exit Criteria:

- All tests in the CI/CD pipeline pass.
- No critical or high-severity bugs/defects indentified in staging.

Reporting and Documentation

- Use tools like Cypress Cloud, Allure or Mochawesome for generating visual test reports
- Document each test phase, results, and any deviations for record-keeping.
- Report defects and bugs in JIRA

Post-Deployment Validation

- Run automated smoke tests in production
- Monitor logs for errors or anomalies
- Collect user feedback for any issues not caught in testing

Testing Flow

1. **Requirement Analysis**: Understand key features, deployment processes, and success criteria.
2. **Test Environment Setup**: Prepare QA, staging, and production environments.
3. **Test Case Design**:
    - Design cases for each test type (functional, performance, etc.).
4. **Execution and Reporting**:
    - Execute tests for each deployment stage.
    - Log and triage bugs, if any.
5. **Final Validation**:
    - Perform a production readiness check with final regression, smoke, and performance testing.
6. **Deployment**: Deploy to production and monitor post-deployment.

Testing Timelines

Phase	Start Date	End Date	Duration	Assignee
Test Case Design	01/12/2024	04/12/2024	3 days	QA Engineer
Test Automation Setup	05/12/2024	10/12/2024	5 days	QA Engineer

Document Review

Document Name	Review Date	Reviewed By	Comments	SignOff
Test Plan	01/12/2024	QA Lead, Product Owner		

Conclusion

This comprehensive test plan ensures that the endpoints are tested across functional, performance, and security dimensions before deployment to production.