CloudNotePad

Test Strategy

Revision History

Date	Version	Author	Description

Table of Contents

1. Scope

In Scope

1. Functionality Testing

- Note Editing: Ensure that the user can write on the notepad
- Note Deletion: Ensure that the user can delete the written note
- Note Saving: Validate that the user can save the created note
- Note Retrieval: Ensure that the user can access the saved note

2. Compatibility Testing

- Mobile Platform: Test the app in IOS and Android based operating systems and verify compatibility with the newest OS release
- Web Browser: Test the application on Chrome, Edge, Safari and Firefox and verify compatibility with the newest updates
- Desktop Platforms: Test the application on MacOS, Windows and Linux, and verify compatibility with the newest OS release

3. Usability Testing

- Evaluate the feel and look of the application, easy of use across all the platforms
- Examine the flow of creation, editing and deletion of the notes

4. Performance Testing

- Response Time: Calculate the time taken to create a new note, edit a note or delete a note
- Lags: Ensure there are no lags and the edited text is visible immediately
- Text Size: Ensure that the notes application is able to handle a big text

Out Of Scope

- No advanced features such as notes collaborations and cloud synchronization
- Performance under load

2. Test Approach

Functional Testing

- Test each functionality across all platforms.
- Verify note creation, editing, deletion, saving, and retrieval.

Compatibility Testing

- Test the application on different devices, browsers, and operating systems to ensure consistent behavior.

Usability Testing

- Conduct usability tests to assess feel, look and user-friendliness of the interface.

Performance Testing

- Measure response times for note operations under normal conditions.

3. Test Environment

- Mobile: IOS and Android latest versions
- Desktop: MacOS, Windows, Ubuntu latest versions
- Web: Chrome, Firefox, Safari, Edge latest and 1 generation older versions

4. Testing Tools

1. Manual Testing

- IOSSimulator: for testing ios apps on MacOS
- Android Emulator: for testing android apps on various platforms
- Google Chrome Developer Tool:
- Mozilla Firefox Developer Tool:
- Safari Inspect
- Manual testing on physical devices: for Desktop version of the app on Linux, MacOS and Windows

2. Automated Testing

- Appium: Appium is an open source automation tool for running scripts and testing native applications, mobile-web applications and hybrid applications on Android or iOS using a webdriver
- Selenium WebDriver: Open-source tool for automating web browsers across different platforms. Can also be used for automating testing of desktop applications on Windows, macOS, and Linux.

5. Release Control

Test Environment Setup:

- Ensure that the test environment is set up with the latest version of the application and appropriate testing tools.

Test Planning:

- Review the test strategy to understand the scope and testing approaches.
- Plan the testing activities based on the test strategy.

Test Execution:

- Execute the planned test cases across mobile, web, and desktop platforms.
- Conduct manual testing and automated testing as stated in test strategy.
- Log defects and issues found during testing.

Defect Resolution:

- Developers address and resolve the reported defects based on their priority and severity.
- Conduct regression testing to ensure that defect fixes do not introduce new issues.

Validation and Approval:

- Once testing is complete and defects are resolved, the testing team validates the application to ensure it meets the acceptance criteria.

Release:

- Once approved, the new version of the application is deployed.

Monitoring and Feedback:

- Monitor the application in the production environment to ensure stability and performance.
- Gather feedback from users to identify any issues or areas for improvement.

Documentation:

- Document the release process, including test results, defects found, and approval details.
- Update the test strategy and other relevant documentation based on lessons learned from the release

Continuous Improvement:

- Review the release process after each deployment to identify areas for improvement.
- Incorporate feedback and lessons learned into future releases to enhance the release control process.

6. Risk Analysis

- Data Loss: The application can have system crushes under certain conditions and cause data loss, can happen from internet connection loss or sending too many requests at once.
- Platform to Platform Differences: The application may act in a different way in 2 different platforms (mobile, web and desktop). Sizes of the components on the UI may differ and sometimes cause application unusability from merged components.
- Performance Issues: The application may have performance bottlenecks under certain conditions for e.g. copy pasting a very large text.