Code and Security Testing Overview

Functional Testing

Functional testing will verify that the software functions the way that it was intended. Also checking to see if there are any errors when navigating through the app. Functional testing will lastly check if all of the user's demands are met and at a point add more features that can enhance user experience or meet additional requirements.

Performance Testing

Performance testing will evaluate the system's speed and stability when testing with example data. It will check to see if the time it takes to generate data is consistent and stable. Also will use performance testing to see the limits of our system and see if it can handle users' demand.

Security Testing

Security testing is used to identify any vulnerabilities or weaknesses in the software code and controls. It involves evaluating the system's ability to protect data, prevent unauthorized access, and be able to handle any unplanned cyber threats. There are many ways to test for security but since our design is fairly simple and straightforward and only checks through the user's functions.

Code and Security Testing Plan

Below is an Excel spreadsheet for testing of multiple google accounts that were used to create multiple tasks and check if all tests passed and to check the expected and actual result:

Users(Google Accounts	Input	Functional Testing	Performance Testing	Security Testing	Expected Result	Actual Result	Pass
User1 (Personal Email)	5 Events on the same day	All functions work properly	Took about 1-2 seconds to process events	No cyber threats are possible	List of events in order	All Events in order of times throughout day	Pass
User2 (Second Email)	10 Events on different days	All functions work properly	Took about 2-3 seconds to process events	No cyber threats are possible	List of events in order	All Events in order of date throughout month	Pass
User3 (Burner Email)	20 Events w/ some on the same day and others on different	All functions work properly	Took about 4-5 seconds to process events	No cyber threats are possible	List of events in order	All Events in order of date and time throughout month	Pass

Code and Security Testing Results

The results of the test were fairly accurate due to the small number of features available to the user and also the simplicity of the design. For functionality testing we had little to no problems with all the classes in our project working as intended and without any errors. For performance, it seems that the efficiency isn't what we had hoped for since it didn't take many events for the DoublyLinkedList to begin taking longer to process. Another issue we discovered was the adding events feature that would actually need all the events to be sorted and then processed through each element to find the correct position to be added in our DoublyLinkedList. So at a point, their data structure will need to be switched to something that can be searched through more efficiently. Besides this, because of the simplicity of our design, we had no issues with security testing since when running we couldn't find a way to breach any data. So overall these results while these results were fairly accurate, there are areas for improvement, especially the performance aspect. However, our current design allows smooth user navigation and minimal security vulnerabilities.

Maintenance overview

Maintenance for our system involves various tasks aimed at ensuring its functionality, security, and efficiency. The types of maintenance required include:

1. **Software Updates and Patch Management:**
    - Regular updates to the GoogleAPI class to keep pace with changes in the Google Calendar API and to address any bugs or vulnerabilities. Patch management ensures that the system remains secure and functional.
2. **Data Parsing and Formatting:**
    - Continuous maintenance of the ParseData class to ensure accurate parsing of JSON data retrieved from the GoogleAPI class. This involves updating parsing algorithms to accommodate any changes in data structure or format.
3. **Event Management and Storage:**
    - Maintenance of the Event class and DoublyLinkedList data structure to handle event creation, storage, and retrieval. This includes optimizing data storage and access methods to improve efficiency.
4. **Graphical User Interface (GUI) Maintenance:**
    - Regular updates to the GUI class to enhance user experience and functionality. This involves adding new features, improving user interfaces, and addressing any usability issues reported by users.

5. **Error Handling and Security Measures:**
    - Ongoing maintenance to implement error handling mechanisms and security measures to protect user data and ensure system reliability. This includes adding try blocks, if statements, and other error detection and recovery techniques.
6. **Documentation Updates:**
    - Regular updates to system documentation, including maintenance plans, and technical specifications. This ensures that users and developers have access to up-to-date information about the system and its maintenance requirements.
7. **Performance Monitoring and Optimization:**
    - Continuous monitoring of system performance metrics to identify performance bottlenecks and areas for optimization. This involves optimizing algorithms, data structures, and system configurations to improve overall performance.
8. **Quality Assurance and Testing:**
    - Ongoing quality assurance testing to identify and fix any bugs or issues in the system. This includes both manual and automated testing to ensure that the system meets functional and performance requirements.

Maintenance Timeline

Maintenance Timelines:

1. **Routine Updates and Patch Management:**
    - Frequency: Monthly
    - Importance: Critical
    - Description: Ensure that the GoogleAPI class stays up to date with any changes or updates to the Google Calendar API. This may include implementing new features, addressing security vulnerabilities, or fixing bugs.
2. **Hardware Inspections and Upkeep:**
    - Frequency: Bi-annually
    - Importance: Medium
    - Description: Perform a thorough inspection of the hardware components used for running the system. Check for any signs of wear, damage, or malfunction. Replace or repair components as necessary to maintain optimal performance and reliability.
3. **Data Backup and Recovery:**
    - Frequency: Daily
    - Importance: Critical
    - Description: Ensure that data backups are performed daily to prevent loss of critical information in case of system failures or data breaches. Test data recovery procedures regularly to ensure their effectiveness.
4. **Performance Monitoring:**
    - Frequency: Continuous
    - Importance: High
    - Description: Continuously monitor the performance metrics of the system, including response times, resource utilization, and user experience. Identify any bugs or inefficiencies and optimize the system accordingly to ensure smooth operation.
5. **Vulnerability Assessments:**
    - Frequency: Quarterly
    - Importance: Critical
    - Description: Conduct regular security audits and vulnerability assessments to identify and address any potential security threats or vulnerabilities in the system.
6. **User Training and Support:**
    - Frequency: As needed
    - Importance: Medium

    - Description: Provide ongoing user training and technical support to address any issues, queries, or training needs that may arise. Keep users informed about new features, updates, and best practices for using the system effectively.

End of Support:

- The end of support for the system is scheduled for 3/4/26. Beyond this date, official support for the system, including updates, patches, or technical assistance, will no longer be provided.

End of Life:

- The end of life for the system is projected for 3/4/28. After this date, the system will no longer be actively maintained or supported, and users will be encouraged to migrate to alternative solutions.