Assignment 1 – Calendar System for Company X

SWE-6613 – Spartans

Names:

- Jeff McGurk
- Elliotte Wideman
- Esther Baah
- Hunter Blake
- Mekonnen Kindo

Part 1

(HB) Accessibility

Requirement:

Users must be able to access the software while in the office and when they are at home.

Description: Maintaining a high level of accessibility is a key component of the company calendar. Company X has a central office location where most of the company operates. However, eligible employees are also allowed to work remotely from their home offices. As such, employees will need the ability to log in, manage meetings, and track invitations while working within the company network and remotely. This also means that the system must be designed with the ability to accept traffic from locations outside the company's network. Proper security protocols will need to be established to ensure connections to the calendar system are authentic and unauthorized access is prohibited.

Measurement: The fulfillment of this requirement will be measured using two tests. First, an automated remote service will periodically check the connection status to the system from outside the company network. This will help simulate typical user operations from a remote site. The system will have a validation goal set at 99% uptime, meaning that at least 9 out of 10 connection requests must be successfully received. Additionally, performance testing tools can be used to further ensure that remote users can access the system without interruptions. Second, real feedback from remote users will be gathered through periodic surveys, asking users to rate their satisfaction with the system's accessibility. The average survey scores will be analyzed to determine if users are satisfied with the system's accessibility from remote locations.

(HB) Scalability

Requirement:

The system must be able to accommodate the removal and addition of new users.

Description: A significant amount of overhead on the system will be attributed to user management. Employee accounts within the system can function individually or as members of groups. This means both individual user accounts and groups can receive or manage calendar invitations. Furthermore, employees can belong to multiple groups, and groups can also have subgroups beneath them. Administrators will require the ability to create, update, and delete user accounts and groups as necessary.

Measurement: To measure the fulfillment of scalability, periodic stress tests will be performed on the system. These stress tests will simulate the creation of 1,000 new test user accounts, each with unique group memberships and roles. Errors or system malfunctions during this mass user creation process will be noted. Afterward, the accounts will be deleted from the system, and any issues arising from this action will also be monitored. These tests will ensure that the system can handle both the addition and removal of users without negatively impacting performance.

(JM) Usability

Requirement:

There will be significant traffic to the system from both internal and remote users. The system must balance the load of simultaneous users.

Description: 60% of the workforce is expected to work outside the office (home or other) at least three days per week. The system needs to support the same set of features, with the same usability and latency, whether the user is in the office or remote. The client system should seamlessly transition between remote and on-site modes without user intervention.

Measurement: The usability of the system will be measured in two parts. First, feature parity will be tested by selecting a group of beta users, divided between in-office and remote usage. These users will provide feedback on the system's performance in both environments. Second, performance monitoring tools, similar to those used for responsiveness, will be deployed from a hosted location outside the company perimeter. These tools will simulate a large remote user population and allow the team to compare response metrics between remote users and those within the network. By analyzing this data, the system’s ability to balance simultaneous user traffic will be evaluated.

(JM) Security

Requirement:

Users will need to authenticate themselves prior to accessing the system.

Description: Due to the sensitive nature of the content within the system, proper security measures must be in place to prevent unauthorized access. Only properly vetted employees or contractors shall be allowed access by presenting a properly issued LAN ID and password combination. A second factor, such as an authenticator app or SMS confirmation code, shall be used upon initial authentication and every 30 days thereafter, or upon re-authentication if a user logs out for any reason.

Measurement: The security requirement will be measured through several layers of testing. First, authentication protocols will be verified by ensuring that all users must provide both a LAN ID/password and multi-factor authentication (MFA) such as an authenticator app or SMS code during login. Testing will confirm that MFA triggers upon initial login and every 30 days, or when users log out. Additionally, penetration testing will be conducted to simulate unauthorized access attempts, ensuring that the system correctly blocks these attempts. Any suspicious activity will be logged and analyzed for potential vulnerabilities. Real-time monitoring of system logs will be used to verify that updates (such as password changes or security breaches) are accurately reflected in the system without delay. The success of these tests will ensure that only authorized users can access the system and that sensitive data is protected from unauthorized access.

(EB) Compatibility

Requirement:

The system should be compatible with various operating systems and devices. It should also integrate with other apps, such as email and task management tools.

Measurement: Compatibility will be measured by testing the system’s functionality across popular operating systems, including Windows, macOS, Linux, iOS, and Android, along with different browsers. Key metrics, such as loading times, responsiveness, and feature functionality, will be recorded to assess how well the system operates on various platforms. Additionally, user feedback will be collected from daily users regarding the system’s ease of use and satisfaction with cross-platform performance. The system’s integration with other apps will be measured using an API tracker, which will monitor the percentage of successful integrations with external tools, such as task management systems. The tracker will provide insights into the rate of success/failure of each API integration and the average duration of API usage.

(EW) Availability

Requirement:

Employees should be able to access core calendar features even without an internet connection, with changes syncing when reconnected.

Measurement: To measure the fulfillment of this requirement, the system's offline capabilities will be tested across various devices and operating systems. Users will be tasked with accessing and modifying their calendar entries while offline. Once the system is reconnected to the internet, it should automatically synchronize any changes made offline. The time taken for the system to synchronize these changes will be recorded and compared against acceptable synchronization thresholds. Additionally, user testing will be conducted where employees make multiple changes, such as adding or modifying appointments, while offline and then reconnect to verify that synchronization occurs without data loss or conflicts. Automated testing tools will be used to simulate network disruptions and recovery scenarios, ensuring the offline functionality and synchronization process work as expected.

(EW) Reliability

Requirement: Track how successfully users can use the system offline. Measure data synchronization time after reconnecting to the internet.

Measurement: Test scenarios will be set up where users perform common tasks, such as adding or updating appointments, while offline. Once users reconnect to the internet, the synchronization time will be measured to track how quickly the system updates with all the offline changes. Monitoring tools will track whether data syncs within an acceptable timeframe, such as 30 seconds. Additionally, data integrity will be checked to confirm that no changes are lost during synchronization. User testing will gather feedback on how easy it is to use the system offline and how quickly and accurately data syncs once they reconnect to the internet.

(EB) Maintenance

Requirement: The system's design should allow new updates and maintenance easily. All documentation used must be able to support troubleshooting and updates.

Measurement: To measure the effectiveness of the system's maintenance, the frequency of maintenance requests will be tracked, along with how easily new bug fixes or features can be deployed. The system's crash rate and the time it takes to repair these issues will also be recorded, helping determine the Mean Time to Repair (MTTR). A lower MTTR indicates less downtime and faster recovery from issues. Easy-to-read user manuals and maintenance documentation will be provided to users, allowing them to resolve simple issues on their own. Feedback from users on the accuracy of the documentation and the ease of fixing issues will also be gathered to ensure the effectiveness of the maintenance process.

(EW) Functionality

Requirement:

The system must remain functional and responsive during peak usage periods. The system needs to offer a summary every week that separates appointments created by users.

Measurement:

To measure the system's functionality during peak usage, tests simulating heavy traffic will be conducted. Tools such as JMeter or LoadRunner can simulate large numbers of users attempting to use the system simultaneously. During these tests, the response time of the system when users attempt to book meetings or modify appointments will be tracked. The system's target response time is under 2 seconds for 95% of peak usage periods. If the system fails to meet this goal, adjustments to server settings or load balancing will be made. Data from these tests will provide insight into the system's performance under stress.

(EW) Performance

Requirement: The system should maintain 95% performance, with response times of under 2 seconds, even during peak loads.

Measurement: The performance of the system will be monitored using tools such as NewRelic or Dynatrace, which will track how long it takes for the system to complete tasks such as adding an appointment or updating a meeting. These measurements will be recorded over time, especially during peak usage periods. The goal is for 95% of tasks to be completed within 2 seconds. If tasks consistently take longer than this threshold, an investigation into the cause of the slowdown will be conducted, and necessary adjustments will be made to improve performance.

(MK) Reliability (Recoverability)

Requirement:

The system should remain operational and recover from failures quickly without data loss.

Measurement:

Reliability will be measured by simulating system breakdowns, such as server failures, and monitoring how long it takes for the system to return to full operation. Automated recovery processes will be tested to ensure that the system can recover smoothly without user intervention. The goal is to restore full functionality as quickly as possible, with no loss of user data during the recovery process.

(MK) Functionality (Suitability)

Requirement:

The system should have functions that accommodate users’ scheduling and event planning requirements, such as making appointments and keeping track of them in a user-friendly and efficient way.

Measurement:

Functionality will be measured by comparing the system’s features against predefined requirements. User testing will be conducted to ensure the system meets users' scheduling needs and aligns with the organization’s goals. Satisfaction surveys will be used to gather feedback from users on how well the system supports calendar management and whether it enhances meeting coordination.

Part 2

Reflection on ‘(NFRs)-in-practice’

Non-functional requirements (NFRs) and functional requirements (FRs) are often distinguished by their primary concern. NFRs specify how a system accomplishes its objectives, while FRs specify what the system does. However, the authors challenge this distinction, highlighting ambiguity between these categories. Their analysis reveals that approximately 75% of the 530 NFRs analyzed in their study described system behavior, which is generally associated with FRs. The findings show that NFRs, despite being considered separate or more abstract, often play a role similar to FRs. They can be elicited, described, and evaluated similarly to functional requirements, mainly because they usually define behavior over a system's interface or architecture. Such a realization is crucial, as it challenges the typical approach in software engineering that addresses NFRs independently from functional requirements.

Reflecting on the authors’ conclusions, it becomes evident that the standard practice of treating NFRs and FRs as distinct entities can create gaps in the development process, potentially leading to incomplete testing and analysis. NFRs, though classified differently, have concrete implications for system behavior, and their separation from FRs can lead to overlooked issues and increased maintenance costs. The authors further emphasize that the lack of precise descriptions and classifications for NFRs increases the problem, as many functional requirements are mislabeled as non-functional, leading to uncertainty in software development and analysis processes. As a result, their conclusion emphasizes the need to reconsider the distinction between FRs and NFRs, supporting a more integrated approach to addressing software requirements.

We strongly agree with the authors’ conclusions. Integrating NFRs into the same frameworks as FRs would streamline development and ensure that crucial system behaviors are not overlooked. The approach avoids the common issue of NFRs being overlooked throughout development, which occurs regularly in existing practices. Incorporating NFRs into the same framework as FRs helps improve traceability, progress control, and the entire validation and verification process, resulting in a more thorough and efficient development cycle.

Works Cited

Eckhardt, Jonas et al. “Are 'Non-functional' Requirements Really Non-functional? An Investigation of Non-functional Requirements in Practice.” *2016 IEEE/ACM 38th IEEE International Conference on Software Engineering*, 2016.