Project Milestone 1: Comprehensive Requirements, Tools, and Technologies

Project: Research Lab Equipment Booking System

Team: Johnjacobjames Erdman, Cole Logan, Shaylin Smith, Asa Fischer

Functional & Non-functional Requirements

Functional Requirements

User Registration – Ensure user authentication and RBAC

Allow users to register and create accounts. This is functional because it requires user action. Contributes to the project by allowing new accounts for the system.

User Verification – Prevent unauthorized access

Allow users to be vetted and accepted/denied based on credentials. This is functional because it is an authentication system for user input. Contributes to the project by only having authorized access to accounts.

Login and Logout – Maintain session security

Allow users to log in and log out with their credentials. This is functional because it requires user action. Contributes to the project by allowing basic session security for accounts.

View Equipment Details – Help users make informed decisions

Allow users to view all the details of any specific piece of equipment. This is functional because it requires user action. Contributes to the project by creating a view for information.

Search Equipment – Make process of finding equipment simple and easy

Allow users to query the list of equipment to find certain equipment. This is functional because it requires user action. Contributes to the project by creating a search system for the database.

Reserve Equipment – Efficient equipment management and usage scheduling

Allow users to reserve a specific piece of equipment for a certain period. This is functional because it requires user action. Contributes to the project by allowing user action on the database.

Cancel Reservations – Ensure flexibility and frees up equipment

Allow users to cancel reservations they have made if they change their mind. This is functional because it requires user action. Contributes to the project by adding another user action on the database.

Usage Logs – Track equipment usage

Allow users to see the availability of equipment and who is using it. This is functional because it requires user action. Contributes to the project by creating a view for information.

Admin Approval – Ensure controlled and authorized equipment use

Allow users to ask for admin approval for the use of equipment. This is functional because it requires admin action. Contributes to the project by creating an admin action on the database.

View Usage Analytics – Provide usage patterns and improve resource management

Allow users (admins) to view patterns and trends based on analytics. This is functional because it requires user action. Contributes to the project by creating a view for information.

Equipment Maintenance Schedule – Ensure equipment is only available when it is in working condition

Allow users to view the status of equipment to ensure they work. This is functional because it is a check on the database. Contributes to the project by limiting parts of the database for certain user actions.

User Profile Management – Ensure accurate user information

Allow users to update their profile to keep their information up to date. This is functional because it is a check on the database. Contributes to the project by having validation of information.

Notification System – Keep users informed (UP FOR DEBATE)

Allow users to subscribe to notifications to stay up to date on equipment. This is functional because it is a dynamic system that will give information to the user. Contributes to the project by having a system that reacts based on database changes.

Booking Confirmation – Provide confirmation notifications (UP FOR DEBATE)

Allow users to give phone number/email for confirmation notifications. This is functional because it is a dynamic system that will give information to the user. Contributes to the project by having a system that reacts based on database changes.

Non-functional Requirements

Scalability – Ensure system can grow with demand

The system will be able to handle increasing usage without issue. This is non-functional because it describes how the system should scale. It contributes to the project because it encourages keeping good performance for larger databases.

Performance – Smooth and efficient user experience

The system will run smoothly and efficiently on all fronts. This is non-functional because it describes how well the system should perform. It contributes to the project because we should keep performance in mind at all times.

Security – Protect user data and system integrity

The system will be secure and not allow user data to be tampered with/seen. This is non-functional because it describes the security of the system. It contributes to the project because we have account data that needs to be secured from external sources.

Reliability – Minimize downtime and ensure consistent availability

The system will be available whenever a user may need it. This is non-functional because it describes the system's resistance to crashes and attacks. It contributes to the project because we want to minimize issues that the system could have.

Maintainability – Reduce maintenance cost and effort

The system will be easily maintained, needing little supervision. This is non-functional because it describes the ease of changes that can be made to the system. It contributes to the project because we want the code to be easily changed in the future.

Usability – Enhance user satisfaction and reduce learning curve

The system will be easy to use and simple to learn. This is non-functional because it describes how easy the system is to use. It contributes to the project because the project has an audience of users that would use the system.

Compatibility – Provide consistent user experience across devices

The system will be uniform across different devices. This is non-functional because it describes how consistent the system is across different systems. It contributes to the project because it has an audience of users that could use the system on different devices.

Documentation – Facilitate system understanding and support

The system will be thoroughly documented for future reference. This is non-functional because documentation is outside of the system. It contributes to the project because it allows for understanding for people that did not work on the project.

Modularity – Simplify development and enhance flexibility

The system will be developed in a modular manner for easier development, testing, and maintenance. This is non-functional because it describes the development of the system. It contributes to the project because we need to keep development simple and flexible for a group of people developing.

Auditability – Provide transparency and accountability

The system will keep logs to support audits. This is non-functional because it describes keeping logs for the project. It contributes to the project because it maintains contributions that each team member makes during the project.

Tools & Technologies

Development Tools

We will primarily use Visual Studio Code as our IDE because it has support for multiple programming languages. VS Code also has a built-in Git integration, which will help streamline the version control within the IDE, instead of using a Git command line. A majority of our group is already familiar with Visual Studio Code, and with how to add extensions. The extensions for our programming languages are well supported, which will be essential for our project.

We will be using SQL, Python, HTML, CSS, and JavaScript for our programming languages. SQL will be used to help us query our database and ensure that we can fetch both user data and equipment data. Python is a flexible language that has multiple free libraries that can help us set up our app and integrate our SQL database. HTML, CSS, and JavaScript

allow us to create a user interface that is easy to use and maintains our functional requirements.

For our frameworks, we will be using React.js for our front-end development and Flask for our back-end development. React.js will be able to help us build intuitive user interfaces that allow the user to perform all the functional requirements. Flask will help us with our back-end development as it allows us to set up our HTML pages and integrate our Python code that will integrate our SQL into our app.

Version Control System

We will be using GitHub for our version control for the project. GitHub is the best fit for our group and the project because everyone in our group is familiar with GitHub. The group's familiarity with GitHub will minimize the time spent learning how to use GitHub. GitHub also has a few nice features that other version control systems do not have. GitHub has an issue system and a Kanban board that makes it easier to define and divide the work for the project. GitHub is also integrated into Visual Studio Code, which streamlines Git commands for anyone that does not like to use the Git command line.

Deployment & Hosting Technologies

Amazon Web Services (AWS) is our choice for a cloud platform because it has a free tier that allows enough resources to host this project. It is the best fit for this project because it has integration support for multiple types of databases like the Oracle Database that we will be using. It is also a reliable service, since it's hosted by Amazon themselves. AWS will support our project's features by hosting our code and database.

We will be using the Oracle Database as our relational database. Oracle is the best fit for this project, because the project requires the use of PL/SQL, which is an extension of SQL made for oracle databases. AWS also has support for Oracle Databases, which will allow easy integration between the cloud platform and relational database. Oracle Database will support our project's features because their requirements are based on the use of PL/SQL, which would require the use of Oracle Database. There will also be a level of familiarity since Oracle Live SQL is based on Oracle Database.

Our client-server model is a three-tier architecture. This is the best fit for the project because there is a frontend, backend, and a database. The front end is the user-interface that the user will interact with. The backend handles any logic needed for the user like login authentication and communicates with the database. The database will store information that the system will require. A three-tier architecture will support our project by clearly dividing the architecture of the project for flexibility and maintainability.