**Technical Specifications**

**For LabEMS**

**2025**

Table of Contents

[1. Introduction 3](#_Toc207215253)

[**1.** **Purpose** 3](#_Toc207215254)

[**2.** **System Context** 3](#_Toc207215255)

[2. System Architecture 4](#_Toc207215256)

[**1.** **System Context** 4](#_Toc207215257)

[**2.** **High-Level Architecture** 4](#_Toc207215258)

[**3.** **UI Layer Decomposition** 5](#_Toc207215259)

[**4.** **Application Layer Components** 5](#_Toc207215260)

[3. Technology Stack 6](#_Toc207215261)

[4. Technical Requirements 7](#_Toc207215262)

[5. Integration Requirements 8](#_Toc207215263)

[6. Testing and Quality Assurance 9](#_Toc207215264)

[**1.** **Unit Testing** 9](#_Toc207215265)

[**2.** **Integration Testing** 9](#_Toc207215266)

[**3.** **System Testing** 9](#_Toc207215267)

[**4.** **User Acceptance Testing (UAT)** 9](#_Toc207215268)

[**5.** **Performance Testing** 9](#_Toc207215269)

[**6.** **Security Testing** 10](#_Toc207215270)

[**7.** **Regression Testing** 10](#_Toc207215271)

[**8.** **Quality Assurance Measures** 10](#_Toc207215272)

[7. Deployment and Maintenance 11](#_Toc207215273)

[**1.** **Deployment Strategy** 11](#_Toc207215274)

[**2.** **Initial Rollout** 11](#_Toc207215275)

[**3.** **Backup and Recovery** 11](#_Toc207215276)

[**4.** **Monitoring & Logging** 11](#_Toc207215277)

[**5.** **Maintenance Plan** 11](#_Toc207215278)

[**6.** **Support & Service Levels** 11](#_Toc207215279)

[**7.** **Future Scalability** 12](#_Toc207215280)

# 1. Introduction

### **Purpose**

The Laboratory Equipment Booking & Maintenance System (LabEMS) is being developed to replace the manual methods currently used to manage laboratory resources, such as emails, spreadsheets, and paper logs. The purpose of the system is to provide one central platform where laboratory equipment bookings, maintenance schedules, and usage records can be managed more effectively. This will make day to day operations faster and more accurate, while also helping the university meet compliance and reporting requirements. The system will also give admin a clear overview of how laboratory resources are being used, which supports better planning and decision making.

### **System Context**

LabEMS will function as a web-based application within North-West University’s existing IT environment. It will be connected to the university’s secure network and authentication systems so that admin and students can log in with their university accounts. The system will make use of the cloud services to store and process data, while also linking with existing tools such as reporting systems and calendars where needed. By fitting into the current IT setup, LabEMS will reduce duplicate work, replace paper-based processes, and give administrators a single, easy to use system for managing bookings and maintenance tasks.

# 2. System Architecture

### **System Context**

A screenshot of a diagram

AI-generated content may be incorrect.

### **High-Level Architecture**

A screenshot of a computer screen

AI-generated content may be incorrect.

### **UI Layer Decomposition**

A screenshot of a cell phone

AI-generated content may be incorrect.

## 

### **Application Layer Components**

Screens screenshot of a phone

AI-generated content may be incorrect.

# 3. Technology Stack

|  |  |  |
| --- | --- | --- |
|  | **Technology** | **Description** |
| **Frontend** | HTML, CSS, JavaScript | This will provide for the structure and content, design and layout, and the behaviour and interactivity of the project. |
| **Backend** | C# (ASP.NET) | API architecture |
| **Database** | MySQL | Relational Database Management System. |
| **Hosting & Deployment** | Azure | Scalable cloud computing platform, cloud services and highly secure data centres. |
| **Authentication** | JWT | Used for authenticating |
| **Version Control** | Git with GitHub | For collaboration control |
| **Dev + Ops** | Docker | Allows you to build, test and deploy applications quickly on any OS. |

# 4. Technical Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| **Serviceability** | Changes and upgrades to the system will not require total outages. |
| **Availability** | The system will maintain an availability of 90% |
| **Performance** | The system will have an average page load time of 2 seconds and under. |
| **Standards** | The system will adhere to our defined architectural and security requirements |
| **Operating system Compatibility** | The system will be compatible with major operating systems to ensure deployment flexibility |
| **Authentication** | The system will be designed to align with our authentication and authorization standards. |
| **Throughput** | The system will handle 200 users simultaneously while meeting all performance objectives |
| **Privacy** | Employees will not be able to view users birth dates through the user interface. |
| **System Errors** | System errors will result in an error code that will be communicated to the user. This code will be well documented in the help desk system to expedite support and incident resolution |
| **Backup and Recovery** | The system will have automated backup routines and a recovery plan to restore operations within the defined RPO (Recovery Point Objective) |
| **Device Compatibility** | The system must be able to hold its functionality when accessing the system from different types of devices (Smartphones, desktops, tablets) |

# 5. Integration Requirements

1. **Email:**   
   Emails will be sent to respective users with updates and reminder notifications with included calendar sync via a .ics file.
2. **Data Exports:**EPPlus (Excel) - This is a .Net library for creating, reading and modifying Excel files. This includes features such as formulas, charts, tables and styling. An excel report of all student reservations, including names, dates of booking and booking information will be produced using this library. For administrative needs, this excel file will then be shared, saved or sent by email.
3. **PdfSharp (Pdf):**  
   This library will be used to generate PDF booking confirmation for students, including their names and booking details. The PDF can then be saved, emailed or printed for administrative purposes.

# 6. Testing and Quality Assurance

A multi-layered testing approach will be used to make sure LabEMS satisfies both functional and non-functional requirements:

### **Unit Testing**

* Every frontend component and every backend C# module will be tested separately.
* Focus areas include role-based access control, maintenance scheduling, booking validation logic, and authentication.
* To verify correctness, automated testing frameworks (such as NUnit, xUnit, and Jest) will be employed.

### **Integration Testing**

* Verify that the database, frontend, and backend APIs all work together seamlessly.
* Test the following external integrations: Calendar Sync (.ics file), Email (Mailgun).
* Confirm that reservations and maintenance schedules are updated in real time for several users.

### **System Testing**

* Complete end-to-end processes (such as scheduling maintenance, booking equipment, and creating reports) were evaluated against the specifications.
* Make sure there are no duplicate reservations and that up to 200 users are handled appropriately.

### **User Acceptance Testing (UAT)**

* Carried out with real postgraduate students, instructors, and administrators from NWU.
* Verify accessibility, usability, and that acceptance requirements (such as ≤5 clicks per reservation and ≤2s response times) are fulfilled.

### **Performance Testing**

* Load tests that simulate up to 200 users at once.
* Stress testing is used to assess scalability during periods of high demand.
* Benchmark outcomes in relation to performance criteria that are not functional.

### **Security Testing**

* Checking for vulnerabilities using penetration testing (SQL injection, XSS, session hijacking, etc.).
* Verification of adherence to GDPR, and POPIA regulations.  
   encryption checks and multi-factor authentication.

### **Regression Testing**

* Every deployment is followed by an automated regression suite to make sure no functionality is broken.

### **Quality Assurance Measures**

* GitHub is an example of continuous integration (CI) with automated builds and test pipelines.
* Before merging, code reviews and static code analysis are conducted.
* Using GitHub Issues for bug tracking and test case management.

# 7. Deployment and Maintenance

### **Deployment Strategy**

* Microsoft Azure hosting for compliance, scalability, and resilience.
* For pre-release validation, the staging environment replicates the production environment.
* Procedures for rollbacks in the event that deployment fails.

### **Initial Rollout**

* Limited users (chosen postgraduate students and admin) will participate in the pilot deployment.
* Feedback loop for system improvement prior to a full-scale launch.

### **Backup and Recovery**

* Daily incremental backups of the database and weekly full backups.
* Backups are safely kept in geo-redundant Azure Storage.
* Quarterly recovery drills are held to verify restoration procedures.

### **Monitoring & Logging**

* Performance and error monitoring with Azure Application Insights and log analytics.
* Notifications for unsuccessful transactions, unusual traffic, and outages.
* All administrator actions are recorded in audit logs.

### **Maintenance Plan**

* Quarterly updates, library/framework changes, and dependency upgrades are examples of preventive maintenance.
* Corrective maintenance involves prioritizing bug repairs according to their severity (normal fixes take five working days, while critical patches take 24 hours).
* Adaptive maintenance refers to changes made for expansion, new rules, or evolving university requirements.
* All system modifications are accompanied by updates to the documentation.

### **Support & Service Levels**

* Target availability: ≥ 90% uptime during semesters.
* Mean Time to Restore Service (MTTR): ≤ 4 hours.
* User support through FAQs and documentation.

### **Future Scalability**

* When usage increases, there is a chance to switch from vertical scaling during the initial deployment to horizontal scaling.
* Easy extension to additional NWU faculties or campuses is ensured by the cloud-based design.