Functional Requirements Document

ENSE 477 2019-2020

LabFiz

Muhammad Ahmed . Zain Abedin . Alwin Baby

Version	Description of Change	Author	Date
1.0	Initial requirements	Zain	01 Nov 2020
1.2	Requirement Revisioned	Alwin	01 Feb 2020

Table of Content

1. INTRODUCTION

LabFiz is a web application, designed to simplify lab inspection and safety management in academic settings. LabFiz allows lab inspectors to automate assignment process and track inspections, to stay up to date on inspection requirements as required by their organization.

1.1 Purpose

The purpose of the FRD is to outline requirements and specifications that will be followed in the design of the software system. The overall purpose in mind is to integrate the software system application into training and inspections for labs.

1.2 Scope

Scope of the system varies based on the stakeholders. For LabFiz, the stakeholders are Safety coordinators and the lab inspectors. certain roles are assigned to potential clients who will use the system. Safety coordinators scope involves inspection generation, inspection reviewing, inspection tracking, ticket issuing, etc. For safety inspectors the scope involves conducting lab safety and submitting reports and ticket handling.

1.3 Background

Workplace and lab safety are an important aspect of any institution. however, there are no feasible management solutions available. The opportunity to design and produce a system that achieves these was presented by the safety coordinator of the engineering department. The proposal was considered by team ZAM for the capstone project. This document is produced in order to outline functional requirements for the project as part of the engineering process. The document is used to describe the business requirements set out by the primary stakeholder. Other stakeholders that are being considered are students, lab supervisors, and safety coordinators.

1.4 References

- Code Review
- User Story Requirements
- Business Proposal and Analysis
- Architecture Diagram
- ERD
- Link to GitHub for reference https://github.com/Capstone2019-ZAM/LabFiz

1.5 Assumptions and Constraints

1. 1.5.1 Assumptions

- Providing a hosting service for University of Regina's usage is to be determined. Production environment will be hosted by the adopter unless a business plan is put in advance.
- Search Engine Optimization is not required for the application
- Closed access for team member is required
- The application is developed under open source license, any code produced to achieve user requirements will be not confidential

2. 1.5.2 Constraints:

- **Business Constraints**

- Time: Development will be done incrementally as per ZAM availability. The project needs to be finished at the end of Winter 2020 semester. Depending on how things progress, future MVPs might or might not be considered.
- Safety Standard: Safety rules and regulations that might need to be considered used in the application are to be accordance with stockholder's requirement.
- Data Privacy: Information and data collected and stored with application is to be confidential. Other policies are to be implemented when hosting the application.
- Cost: Cloud based hosting solutions need to be considered that will provide facilities for hosting the application and data.
- Ease of Use: The application needs to be usable in labs by variety of users with varied technical skills.

- **Technical Constraints:**

- Learnability: How efficiently we will be able to understand the syntax and semantics for the technologies that we will be using in the software system.
- Industry Trends: Newer stacks and framework may not be feasible to learn and implement in the given timeframe.

Layer	Technology
Client	JavaScript Framework
Server	Laravel/PHP
Database	MySQL

1.6 Document Overview

The document provides insight into a software system designed to enhance the laboratory experience for students, lab supervisors, and safety coordinators. The software system will be designed with the intention to bring awareness to lab safety rules and regulations in a convenient way.

In this regard, the system can be considered multi-purpose that can offer services to multiple stakeholders' types.

For safety inspectors the system can be used for incident report building and performing inspections.

For students, the system can be used to educate them on standard lab protocols and regulations.

For supervisors, the system can provide real time access to issue tracking and check-list/reminders.

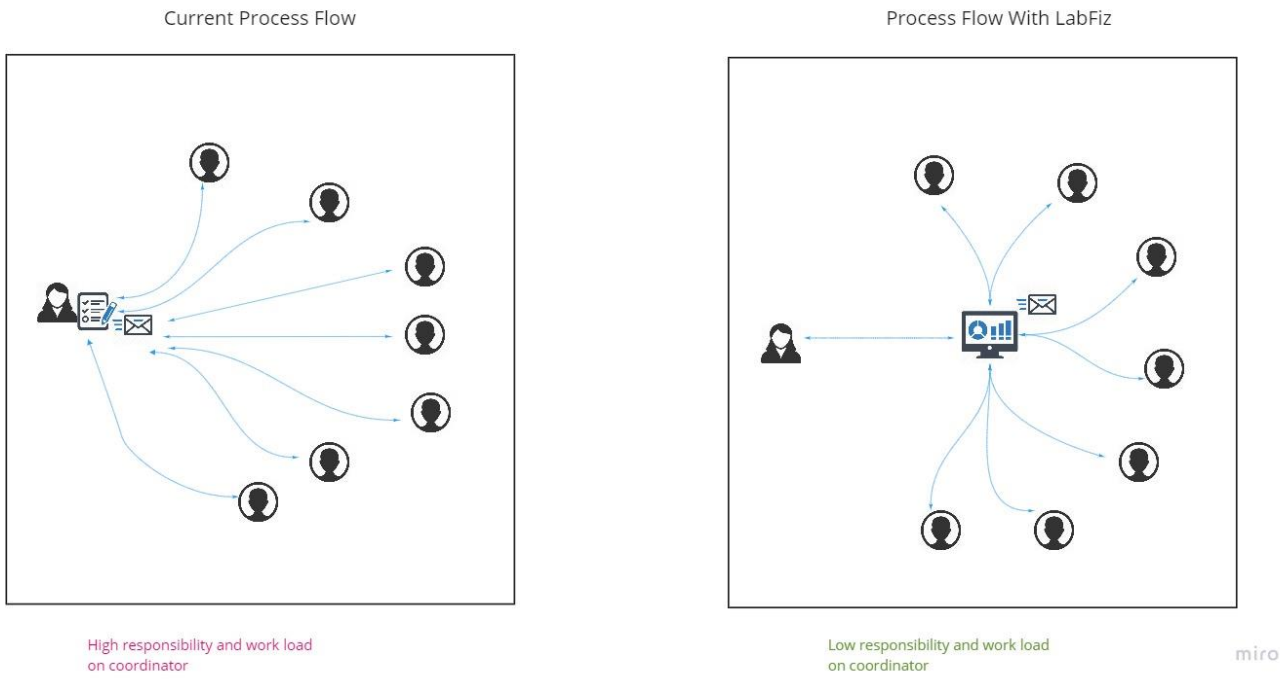
2. Methodology

A standard report reading style approach can be adopted when reading the FRD contents. Technical details are not extensively delved into, but are somewhat explored at a higher level to give transparency into the architecture of the software system without overwhelming the reader.

3. Functional Requirements

3.1 Context

The current and future business process is depicted in the diagram below:



3.2 User stories

1	Safety Coordinator	Able to create and update templates for inspection report	High
2	Safety Coordinator	Able to create instance of a report	High
3	Safety Coordinator	Able to assign a registered Safety Committee member/Lab Supervisors to it	High
4	Safety Coordinator	Able review and approve submitted reports	High
5	Safety Coordinator	Able to track reports	High
6	Safety Coordinator	Able to create and assign "Issue" tickets for labs	High
7	Safety Coordinator	Able to create and delete accounts for Safety Committee members and Lab Supervisors	High
8	Safety Coordinator	Able to send notification email to inspectors	Low
9	Safety Coordinator	Able to ask and reply to questions on a report submission	Low
10	Safety Coordinator	Able to receive reminders for upcoming inspections	Low
11	Safety Coordinator	Able to identify re-occurring items on inspection reports	High
12	Safety Coordinator	Able to get notified about overdue tasks	High
13	Lab Supervisors / Safety Committee members	Able to submit an **assigned** report	High

14	Lab Supervisors / Safety Committee members	Able to track reports	Medium
15	Lab Supervisors / Safety Committee members	Able to create and fill "Issue" tickets	High
16	Lab Supervisors / Safety Committee members	Able to see notification/emails	Medium
17	Lab Supervisors / Safety Committee members	Able to ask and reply to questions on a report submission	Medium
18	Lab Supervisors / Safety Committee members	Should be able to login with an authorized account	High

3.3 Data Flow Diagrams

3.4 Interface Requirements

- While the coordinator assigns an inspection to an inspector, it should dynamically update in the inspector's page. And it should also send an email notification.
- Coordinator should be able to see any reports at any stage.

3.5 Hardware/Software Requirements

- Any web browser can be used to access the webpage
- Webpage can be easily adapted to any screen size without losing any functionality.

3.6 Operational Requirements

- To operate the application, a traditional stack is required. For this, LAMP stack consisting of apache, Linux, MySQL and PHP is used.

- The hosting of the production environment is up to adopter's discretion, however for demo purposes AWS Elastic Beanstalk is used.
- The operational cost associated with higher computing and network traffic limits AWS EC2 instances to free tier for this project.
- Availability and robustness with AWS usage will suffice the need of the demo site.

3.6.1 Security and Privacy

A. State the consequences of the following breaches of security in the subject application:

1. Loss or corruption of data will result in losing all the saved work. There might not be any easy ways to get the data back if the data got corrupted or lost.
2. Disclosure of secrets or sensitive information is a huge deal when it comes to the academic industry. It may result in unauthorized users getting access to the system and can be used for malicious activities.
3. Disclosure of privileged/privacy information about individuals is very important. Disclosure of information about the users is one of the highest priorities. If we fail to do so, the webpage might get banned or might lose trust in most users.
4. Corruption of software or introduction of malware, such as viruses can lead to numerous problems such as losing data, information leak, etc.

3.6.2 3.6.2 Reliability

1. State the damage can result from failure of this system—indicate the criticality of the software, such as:

   a) Since the system is used for lab safety, it is important to make sure that the system doesn't fail. If it fails, there is a possibility that some of the inspection may be wrong and can result in various damages

   b) Loss of time - If the system doesn't work as planes, there is a possibility that someone will have to manually do the inspections which could take a lot of time.

   c) Loss of human life- If the system fails to remind on time and if the safety inspection fails, there is a possibility it might end up in severe damage caused by the lab equipment.

2. What is the minimum acceptable level of reliability?

Mean-Time-Between-Failure -Since the system is hosted by AWS, it should not fail unless the server is down. And AWS is one of the best servers out there and it should not fail.

If the system failed, it should be easy to restart the whole system. It won't take long since all the records will be saved in the database.

3.6.3 Recoverability

A. In the event the application is unavailable to users (down) because of a system failure, how soon after the failure is detected must the function be restored?

   If the system fails it will need to be restored right away. Migrations allow database restore easily in Laravel. However, there is automatic system notification set on the application layer to inform admin. Currently AWS handles it on infrastructure layer. It is adopter's responsibility to ensure system health notifications are set as per hosting service providers guidelines.

B. In the event the database is corrupted, to what level of currency must it be restored? For example, "The database must be capable of being restored to its condition of no more than 1 hour before the corruption occurred".

   If the database is corrupted there are no alternative ways to replace it at the moment without data recovery region setup.

C. If the processing site (hardware, data, and onsite backup) is destroyed, how soon must the application be able to be restored?]

   If the site is destroyed due to any reason, it will be restored to the initial state given that data backups are turned on by the adopter. All the records that are not backed up in the database will be lost and the administrator/inspector will have to redo the inspections.

3.6.4 System Availability

The demo system is available 24/7 through labfiz.com

3.6.5 General Performance

Response time from the servers are fairly quick right now. It might take a few minutes to get a password reset link. Except that the whole performance is really fast.

Since the webpage is designed for university use, the Expected rate of user activity depends on the universities lab inspections.

3.6.6 Capacity

The capacity of the database is also depending on the adopter and their hosting solution. Since universities have huge data storage, the number of entries shouldn't be a problem. If the limit exceeds there is always other ways to back up the data somewhere else.

3.6.7 Data Retention

Since our webpage is used for lab safety the records will need to stay in the system for a long time. it can be deleted as wanted. Since university is using the system, they have the option to keep it as long as they need as per their rention policy.

3.6.8 Error Handling

The system handles error messages efficiently on different layers. The server responds back to the client with appropriate HTTP code such as ERR 404 or ERR 500 when an error occurs. The client side than render the page accordingly to handle user input on errors.

3.6.9 Validation Rules

Validations are carried out at different layers. The client side ensures the data is transmitted and displayed in the correct format.

Laravel's validation layer ensures correct data types and values that meets the coded regular expression are accepted before passed on to the controller layer.

The controller/service layer ensures business logic is made in any CRUD operation. Further rules set in model layer prevent any changes to database that may break fundamental data storage logic.

3.6.10 Conventions/Standards

The application itself adopts MVC architecture. However different technologies for different layer of web application have their specific design.

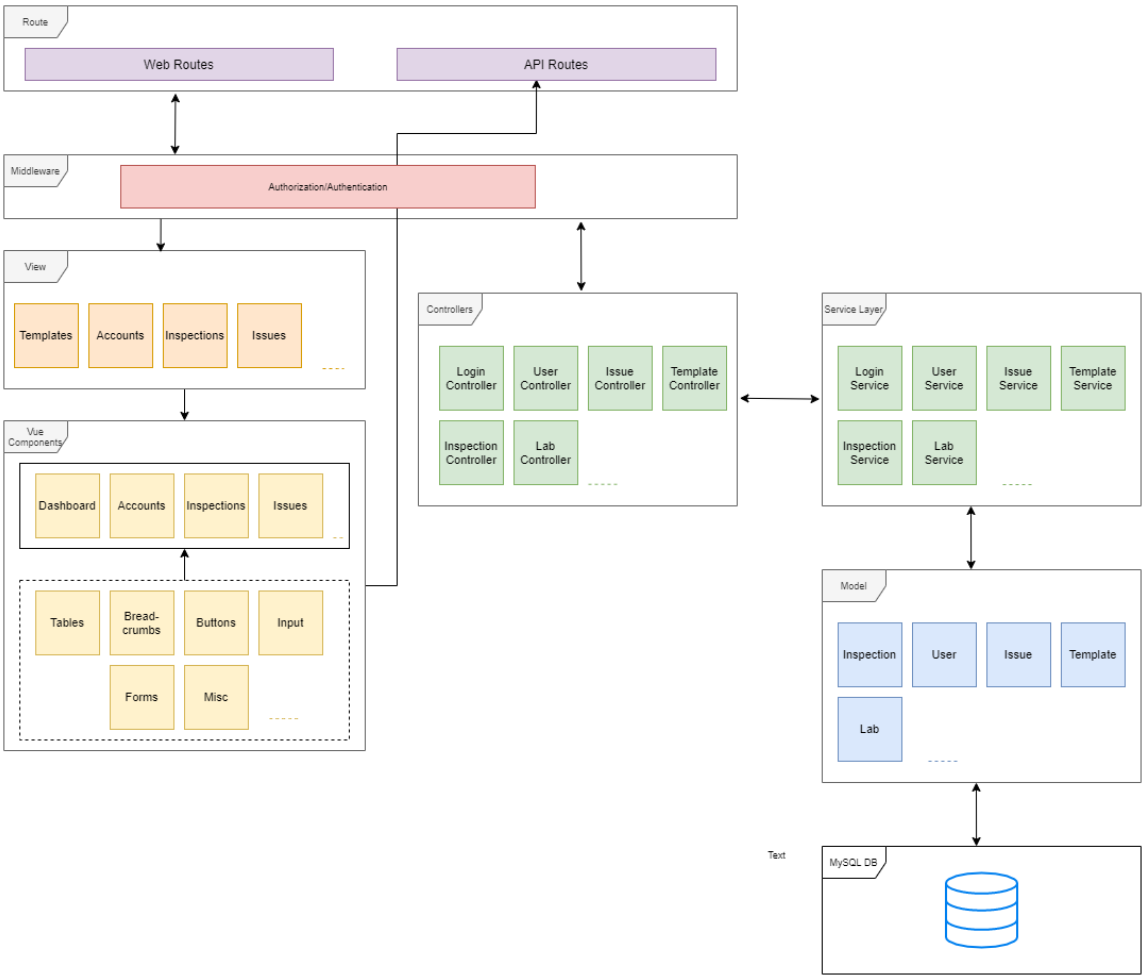
RESTful API convention is used to design API layer for the application.

Front End is designed used component-based approach using single file components. Framework specific coding standard for Vue.js are adopted.

Web Server is designed service layer architecture with repository model interface for database interaction. Laravel Framework specific coding standards are adopted for web server layer.

4. System Architecture

The architecture diagram below details the architecture used for the software system.


Route
Web Routes
API Routes
Middleware
Authorization/Authentication
View
Templates
Accounts
Inspections
Issues
Vue Components
Dashboard
Accounts
Inspections
Issues
Tables
Bread-crumbs
Buttons
Input
Forms
Misc
Controllers
Login Controller
User Controller
Issue Controller
Template Controller
Inspection Controller
Lab Controller
Service Layer
Login Service
User Service
Issue Service
Template Service
Inspection Service
Lab Service
Model
Inspection
User
Issue
Template
Lab
Text
MySQL DB