Requirements of the report management system

By:

Dembegiotis, Callista	s5822637
Nieuwenhuis, Demian	s5163609
Giummarra, Alessio	s5793181
Mura, Maria	s4778898
Willmann, Leo	s5253144

Faculty of Science and Engineering, University of Groningen

WBCS017-10: Software engineering

June 2, 2024

Table of Content

Introduction

This document is one of three parts that will discuss the report management system developed for Gomibo. This document will discuss the requirements of the application.

This project is based on a need of the Groningen department of Gomibo. Gomibo is a technology company that is a part of the Dutch telecom webshop Belsimpel. They offer phones, tablets, gadgets and cellular contracts. At Gomibo there's a maintenance team responsible for anything related to the office. They use a Google Form for any reports related to malfunctions in the office and store the reports in an excel sheet. While the system is fine there are a lot of improvements that can be made related to user experience. Our goal is to develop a report management system that will increase ease of use for both the employees that report malfunctions and for the maintenance team. The maintenance team has multiple locations in close proximity to take care of, therefore we will develop a web-based application such that malfunctions can be reported regardless of location.

The current workflow is as follows. An employee reports a malfunction through the google forum. The google forum asks for email, name, team group, location, malfunction type, inventory ID number of the product, and a description of the malfunction. This information is then put into an excel spreadsheet where the maintenance team assesses the report and addresses the malfunction.

The maintenance team has expressed a desire to have this workflow changed to all happen in a single web application where they can easily see reports, set priority, and manage work distributions together with their colleagues. They also want to have support for images in reports, a dashboard with statistics information, automatic email notifications, and QR codes to the website and possibly to a certain report already filled in with the items information.

Personas

The personas are the people who will use the web service.

- **The reporter:** this is a Gomibo employee or an invited guest that will report malfunctions they encounter at the location the malfunction report system will be deployed. The reporter is interested in having the office equipment fixed as soon as possible to better their working environment.

- **The maintenance personnel:** this is a Gomibo employee that is responsible for fixing the malfunctions. The maintenance personnel want to be able to view the submitted reports and edit select fields to manage them. They also want their access to the system to be granted only after they login. This is to prevent unauthorized users from viewing and editing sensitive reports on malfunctioning office equipment.

- **The developer:** this is a Gomibo employee who will run and maintain the application. They are interested in receiving an application that is using familiar environments. Therefore we will use Laravel for the backend and react for the frontend of the application.

Stakeholders

- **Maintenance team:** the maintenance team of Gomibo is the major stakeholder, we aim to improve their experience compared to the current system and thus help with a more streamlined, efficient and clearer maintenance process.

- **Reporters:** reporters are the second biggest stakeholder. We aim to make it easier, faster and more approachable to report a malfunction compared to the current system.

- **Developers:** will need to set up, configure and manage the hosting of the web application. In addition to managing the deployment, they will also need to maintain the application. Therefore we aim to make this as simple as possible.

- **Other employees of Gomibo / Non-reporters:** they will benefit from a better maintained office, with things working as they should.

- **Building proprietor(s):** our web service will affect the maintenance of the building, thus the proprietor(s) will have an interest in our service.

- **Visitors / Non-employees:** the web service will cause a more efficient maintenance process and result in a more well maintained office. Which is seen also by visitors/non-employees.

Epics

The aim of an epic is to organize tasks in a clear hierarchy. We have one epic, the malfunction report management system. This epic has a list of 3 features that are further broken down into user stories. Each user story (US) follows with requirements to the feature.

Malfunction Report management system

The entire web application that has three main features:

- Report management: *US-1*, *US-3*, *US-4*, *US-5*, *US-6*, *US-8*, *US-9*
- Report submission: *US-2*, *US-7*, *US-10*,
- Deployability: *US-11*, *US-12*, *US-13*

User stories

User stories are used to give the reader an informal and general explanation of the features that are developed in the project. The stories are from the point of view of the personas discussed before.

Each user story has a persona articulating what value a certain feature will bring to the system. A list of testable requirements that each user story produces is then presented, alongside the acceptance criteria that indicates when a story is deemed complete. The requirements are categorized type and on priority with functional / non-functional (F / NF) requirements and the MoSCoW technique respectively. Functional requirements define what the system must do and non-functional requirements detail how it should do so. The MoSCoW technique splits the requirements based on the importance to the client into Must have, Should have, Could have, and Will not have.

- Must have: non-negotiable product needs that are mandatory for the client
- Should have: important needs that are not vital, but add significant value
- Could have: nice to have features that will have a small impact if left out
- Will not have: features that are not a priority for the specific time frame

The requirements ID tag is composed as follows:

US{number}-{Requirement priority}{Requirement number}

Where

- {number} indicates the user story number.
- {*Requirement priority*} is the initial letter from the MoSCoW prioritization.
- {*Requirement number}* is the nth requirement belonging to the user story.

The user stories may refer to the database, backend, frontend and API of the application. More information on these components is available in the design document.

The priority column of the following requirements will indicate that the requirement has been achieved if set to ‘Must’ and not achieved if set to ‘Want’. The requirements ID tag indicates the original set priority.

US-1 Management login

As a maintenance personnel, **I would like** to be able to login to the report management system, **so that** only maintenance personnel can manage malfunction reports.

ID	Requirement	Priority	F / NF
US1-M1	There is a login and authorization system for the role of 'report manager’ whenever a user logs into the web service.	Must	F

US-2 Report submission

As a reporter, **I would like** to report a malfunction, **so that** it gets fixed as soon as possible.

For use case submit report:

ID	Requirement	Priority	F / NF
US2-M1	The API must be able to retrieve the questionnaire from the database.	Must	F
US2-M2	The API must be able to create the report with responses in the database.	Must	F
US2-M3	The reporter must be able to fill in the questionnaire, and submit it.	Must	F
US2-M4	The API must be able to create answers in the database to answer open questions.	Must	F

For report submission with file upload:

ID	Requirement	Priority	F / NF
US2-S5	Reporters should be able to optionally add files they want to include in their submission.	Must	F
US2-S6	The API should store uploaded files and create file entries in the database.	Must	F
US2-S7	The files should be stored in server local storage.	Must	F

US-3 View reports

As a maintenance personnel, **I would like** to be able to view the submitted reports, **so that** I know what equipment needs maintenance.

ID	Requirement	Priority	F / NF
US3-M1	Reports must have a status and time field. The status can be ‘resolved’, ‘in progress’ or ‘not resolved’.	Must	F
US3-M2	The API must be able to retrieve a single report with all its responses from the database.	Must	F
US3-M3	The API must be able to retrieve all report entries with search filters such as ‘by priority’ or ‘by status’ from the database through the backend.	Must	F
US3-M4	Only a logged in `report manager` must be able to retrieve reports through the API.	Must	F

When report has files attached:

ID	Requirement	Priority	F / NF
US3-S5	Maintenance personnel should view the file when viewing a report.	Must	F
US3-S6	The API should be able to download file entries from the stored file path in the database.	Must	F
US3-S7	The files should be read from local storage.	Must	F

US-4 Update status and priority

As a maintenance personnel, **I would like** to set the priority, and status of the submitted reports, **so that** fellow maintenance personnel know the priority and status of the report.

ID	Requirement	Priority	F / NF
US4-M1	The API must be able to update report entries’ status and priority in the database.	Must	F
US4-M2	Only a logged in `report manager` must be able to update report entries’ status and priority through the API.	Must	F

US-5 Manage questionnaire

As a maintenance personnel, **I would like** to manage the questionnaire asked in a report submission, **so that** any change in the workspace, equipment or errors in the questionnaire can be addressed.

ID	Requirement	Priority	F / NF
US5-M1	The question must have a list of answers that are presented as multiple choice answers. This list can be empty.	Must	F
US5-M2	The API must be able to create questions with its belonging multiple choice options in the database.	Must	F
US5-M3	The question must have the field `is_open` to indicate if an open answer is allowed.	Must	F
US5-M4	The question must have the field `is_active` to indicate if it is actively used for the questionnaire.	Must	F
US5-M5	The API must be able to update the `is_active` field of questions in the database.	Must	F
US5-M6	Only a logged in \`report manager\` must be able to create and update questions through the API.	Must	F

US-6 Employee assignment

As a maintenance personnel, **I would like** to be able to assign myself and others to reports, **so that** fellow maintenance personnel know who is attending to it.

ID	Requirement	Priority	F / NF
US6-S1	A ‘report manager’ should be able to assign themselves to a report.	Must	F
US6-S2	The API must be able to update report entries in the database to assign ‘report manager’ to the report.	Must	F
US6-S3	Reports must have an assigned maintainer field.	Must	F

US-7 Email notification

As a reporter, **I would like** to know when my reported malfunction is fixed, **so that** I know when the equipment malfunction has been addressed.

ID	Requirement	Priority	F / NF
US7-M1	The questionnaire has a question asking for the email address of the reporter	Must	F
US7-S2	The questionnaire has a question asking to allow email notification when the status of the report is marked as ‘resolved’.	Must	F
US7-S3	The system should send an email notification to the reporter when a ‘report manager’ sets the status of the report as ‘resolved’.	Must	F

US-8 Statistics of reports

As a maintenance personnel, **I would like** to see statistics of the reports, **so that** I can see how many reports have been submitted / resolved over a period of time.

ID	Requirement	Priority	F / NF
US8-S1	A ‘report manager’ should have a dashboard with statistics on how many malfunctions have been fixed over a set timeframe.	Want	F

US-9 QR code generation

As a maintenance personnel, **I would like** to link pre-filled reports to a QR code, **so that** when reporters scan the QR code they have an easier time submitting the report.

ID	Requirement	Priority	F / NF
US9-C1	The API must be able to store pre-filled reports	Want	F
US9-C2	The pre-filled reports could be accessible through a URL.	Want	F
US9-C3	The pre-filled report’s URL can be converted to a QR code within the application.	Want	F

US-10 QR code access

As a reporter, **I would like** to report with pre-filled reports, **so that** some of the necessary information has already been entered.

ID	Requirement	Priority	F / NF
US10-C1	The API could be able to access the pre-filled reports	Want	F
US10-C2	The pre-filled reports could be accessible through a URL.	Want	F
US10-C3	When using the URL to open the pre-filled report, it allows reporters to enter data and submit their report as a normal report submission.	Want	F

US-11 System environment

As a developer, **I would like** to have the system be developed in familiar environments, **so that** any eventual system maintenance or changes do not require the learning of new frameworks or languages.

ID	Requirement	Priority	F / NF
US11-M1	The backend must be developed with the PHP framework Laravel.	Must	NF
US11-M2	The frontend must be developed with the JS framework React.	Must	NF
US11-M3	The API must adhere to the RESTful principles.	Must	NF
US11-S4	The web service should use a MySQL database docker image.	Must	NF
US11-S5	The entire web service must be able to be deployed with Docker.	Want	NF

US-12 Deployment instructions

As a developer, **I would like** to have an instruction set, **so that** it is easier to deploy the system.

ID	Requirement	Priority	F / NF
US12-M1	The system's root directory has a `README.md` file with information and instructions to deploy and manage the system.	Must	F

US-13 Maintenance

As a developer, **I would like** to receive a system that has high code quality, **so that** it is easier to extend and maintain the software.

ID	Requirement	Priority	F / NF
US13-M1	The system has overall a high code quality (judged by the client). Some examples of this are minimal code duplication, high test coverage, and code metrics.	Must	F
US13-M2	The system directory, package, and module structure is easy to follow.	Must	F

Progress

Who	When (2024)	Where	What
All	19 feb	Introduction Epics User stories	Initial document additions
Leo	25 feb	Requirements	Added requirements
Leo	26 feb	Technical requirements	Added section
Leo	26 feb	Epic	Introduction; Format of epics; Features; Many use cases
Demian	3 march	Feature view & edit reports	Added view & edit reports requirements
Callista	3 march	Feature report system	Added information (actor, description, trigger, flow) under both use cases
Alessio	3 march	Features QR code generation	Added information and requirements
Maria	4 march	Features: Login, statistics, view and edit malfunction	Added information and requirements
Callista	4 march	Malfunction Report System Epic	Added requirements for each use case
Leo Demian	4 march	Maintain categories for objects and locations	Added use cases and requirements
Leo	11 March	Epics	Numbered requirements

Leo	26 March	Entire document;
		User stories;
			Followed feedback;
			Introduction and explanation of user stories. Added user stories 1 - 8;

Leo	26 March	Document style	Title page Contents Page number
Leo	28 March	Introduction; User stories	Updated introduction; Added user stories 9 - 11
Leo	3 May	Personas, epics	Added development persona; Changed epics, and features.

Leo	5 May	User stories; Technical requirements	Added user stories 12, 13, and 14. Removed Technical requirements
Leo	13 May	User stories	Added use cases to report submission and view report US to include file upload and access.
Leo	27 May	User stories	Moved acceptance criteria to the testing document
Leo	1 June	User Stories	Updated requirements priority for achieved requirements to ‘Must’ and for not achieved requirements to Want’.