Software Requirements Specification

Document owner: Samuele Pozzani is responsible for developing and maintaining this document.
Date: January 05, 2020
Version: 1.0

Revision History

Version	Date	Name	Description
1.0	01/05/20	Samuele Pozzani	Initial document

1 Introduction

1.1 Overview

MarconiTT is a web application available to desktop and mobile users and accessible through a browser. It can be reached both from inside and outside the school network. The application allows for authorized teachers, students and staff members to book school classrooms and labs. Team 3 is interested in updating MarconiTT, with three important changes and different minor fixes.

This document provides information on the requirements for the MarconiTT application. Project goals, scope and definitions are given in the introduction. Design contraints and application environment are described in the following section. Non-functional requirements are outlined for later verification. Functional requirements are given to show the system features and expected user interaction.

Project constraints will be included in separate documentation. The Software Project Management Plan will give specifics on project budget and schedule. A separate Test Plan document will address test specifications and procedures.

1.2 Goals and Objectives

The overall objective is to add new features that increase usability and to containerize the entire application so that further developments are easier to implement on the production server.

1.2.1 Project Goals

1. Increase user satisfaction by allowing them to perform basic operations quicker
2. Decrease the amount of time needed to update the application
3. Increase overall performance

1.2.2 Project Objectives

1. Update the booking form so that it allows users to book the same classroom for multiple hours on multiple days of the week

2. Update the "Gestisci prenotazioni" section by adding a calendar that allows users to filter previous bookings based on the selected date
3. Create two containers that hold the Database and the Backend application using Docker and Docker-Compose
4. Create a program that can be used to update the timetable on MarconiTT inside the containerized environment
5. Move the database and the backend application inside the Docker containers
6. Move the Webserver with the containerized version MarconiTT on a new server (edu-x08)

1.3 Scope

The new features that are to be implemented will allow users to book classrooms for up to 10 hours for multiple days in consecutive weeks using the same form, as opposed to having to book for each day individually. Users will also be able to see previous bookings filtered by day thanks to a small calendar that will be implemented in the "Gestisci prenotazioni" section, as opposed to having all previous bookings in one list with no filters. The MarconiTT logo will be updated with a new one that will be created using the current I.T.I. G. Marconi logo.

The encapsulation of MarconiTT inside a containerized environment eliminates most of the difficulties that developers encounter when updating a software on production servers, since the code will reside inside an identical virtual environment on both the development and production machines.

1.4 Definitions

MarconiTT - the product that is being described here; the application in which team 3 is adding new features and applying changes
Project - activities that will lead to the production of the new version of MarconiTT
Client - the organization for which this application is being built
ITI G. Marconi - a technical institute in Verona, the client of this application
User - the person or persons who will actually interact with MarconiTT
Use Case - describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes
Scenario - one path through a use case
Actor - user or other software system that receives value from a user case
Developer - the person or organization developing the system, also sometimes called the supplier
Stakeholder - anyone interested in the project and its outcomes. This includes clients, customers, users, developers, testers, managers and executives
Container - a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another
Booking - a reservation of a classroom or a lab for one or more hours in a day
Timetable - a table that describes classrooms occupations for every hour
Backend - parts of a computer application or a program's code that allow it to operate and that cannot be accessed by a user
Frontend - all software or hardware that is part of a user interface. Human or digital users interact directly with various aspects of the front end of a program, including user-entered data, buttons, programs, websites and other features
edu-x08 - the production server that runs Debian 11, docker and docker-compose

2 General Design Constraints

2.1 Application Environment

MarconiTT will be able to run on any OS that has docker and docker-compose installed. The use of multiple containers inside the Docker environment allows developers to develop the app locally and then move it on production servers with no friction. Working with containers orchestrated by docker-compose will also allow for the database to be completely isolated in a way so that only the backend will have access to it.

2.1.1 Development Environment

The development environment is supposed to run locally, on every machine with docker and docker-compose. Docker will run 4 containers:

1. Database (mysql 5.7)
2. Backend (NodeJS)
3. Timetable updater (alpine 3.7 - Python)
4. Frontend (AngularJS)

NB: Frontend container is only needed in the development environment.
Check configuration files to fit system specifications:

- docker_env/angular/app/app.js
- docker_env/node_server/config_default.js
- docker_env/carica_orario/scripts/orario_conf.json

2.1.2 Production Environment

The production environment is supposed to run on edu-x08 production server. Since Apache web server is running separately, the frontend container isn't needed (the AngularJS application should be reached by apache). Docker will run 3 containers:

1. Database (mysql 5.7)
2. Backend (NodeJS)
3. Timetable updater (alpine 3.7 - Python)

2.2 User Characteristics

Authorized users: Only authorized students, teachers and staff members can log in on the application in order to book classrooms and remove existing bookings.
Regular users: They don't need to authenticate, they can see timetable and bookings.

2.3 Mandated Constraints

The application will run on edu-x08 production server. Only a few users will be authorized to log in and create bookings.

3 Non-functional Requirements

3.1 Operational Requirements

Usability: Regular users will not need to read the user manual to be able to use the application. Authorized users will be provided a brief guide that describes the most important features.

3.2 Performance Requirements

Maintainability: The application can be easily updated thanks to Docker and Docker Compose. Developers will be able to make changes to any component locally and then push those changes to the production environment. Each change that is made to any of these components needs to be carefully analysed before being adopted inside a production environment. Not considering how a small change (to a single component) interacts with the other components can lead to multiple issues.

3.3 Security Requirements

MarconiTT application can be reached either from inside and outside the school network. Authorized users can log in on the app using the same credentials used to log in on the school network (username and password). Authorized users will be authenticated through LDAP.

3.4 Documentation and Training

MarconiTT will be installed on edu-x08 production server with a user guide accessible from the application.

3.5 External Interface

The user interface will be eye-catching and visually appealing. When users access their accounts, the interface will provide a custom navigation menu which has a straightforward look and feel. The interface will be intuitive and simple to use. No training will be provided and it is expected that 95% of users will be able to use the app without any training.
The updated interface will allow authorized users to book the same classroom for multiple hours on multiple days of the week in the easiest way possible.

4 Functional Requirements

4.1 Feature: Docker Containerization

4.1.1 Description

Create two containers that hold the Database and the Backend application using Docker and Docker-Compose. Move the database and the backend application inside the Docker containers. Move the Webserver with the containerized version MarconiTT on a new server (edu-x08).

4.1.2 Use Case 1

Description: Application update on the production server
Actors: developer

Value: high
Cost: high

Basic Path

1. Log in on the production server using SSH

2. Obtain super user privileges using:
    - `$ su`
3. Stop docker-compose:
    - `$ docker-compose down`
4. Copy the container folder with the new changes to the docker_env folder on the server
5. Set pruduction configuration on:
    - docker_env/angular/app/app.js
    - docker_env/node_server/config_default.js
    - docker_env/carica_orario/scripts/orario_conf.json
6. Run the following commands into the docker_env folder:
    - `$ docker system prune -a`
    - `$ docker-compose -f docker-compose-prod.yml build`
    - `$ docker-compose -f docker-compose-prod.yml up -d`
7. Disconnect from the pruduction server

4.2 Feature: Automatic Timetable Update

4.2.1 Description

Create a program that can be used to update the timetable on MarconiTT inside the containerized environment. There will be a shared folder in which a script checks the existence of new files every 60 seconds and automatically runs the timetable update script.

4.2.2 Use Case 1

Description: Timetable update
Actors: school staff members

Value: high
Cost: medium

Basic Path

1. Log in on the pruduction server using SFTP (CLI or Filezilla)
2. Copy the updated files GPU001 and GPU004a to `/flussi_marconitt`
3. Await until the updater starts (max 60 seconds)
4. Disconnect from the production server

4.3 Feature: Multiple day booking

4.3.1 Description

Allow an authorized user to create a booking that will be repeated a specific number of weeks.

4.3.2 Example

An authorized user wants to book a classroom every friday for 3 hours. By selecting the number of repetitions it will be possible to book the classroom every friday for *n* weeks.

4.3.3 Use Case 1

Description: Book a classroom for multiple days
Actors: Authorized users

Value: high
Cost: medium

Basic Path

1. Log in on MarconiTT using user credentials
2. Select the day to create the booking
3. Select the classroom and the repetition
4. Select the hour
5. Select for who is the booking
6. Click on 'Conferma'
7. Disconnect from the application