A8 - Vertical Prototype

Databases and Web Applications Laboratory (LBAW)
Bachelor in Informatics Engineering and Computation (L.EIC)

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

Prelims

Discussions

- **Topics discussed in classes and online.**

- How to store files in/with the database?
  - Impact / Solutions
- How to delete user information and keep records?
  - Solutions

EAP: Architecture Specification and Prototype

- This component groups artifacts related to the high-level architecture specification of the information system to be developed and the vertical prototype implemented to validate the architecture.

- **A7: Web Resources Specification**
  This artifact presents an overview of the web resources to implement, organized into modules. It also includes the permissions used in the modules to establish the conditions of access to resources.

- **A8: Vertical Prototype**
  Includes the implementation of the features marked as necessary (with an asterisk) in the common and theme requirements documents. This artifact aims to validate the architecture presented, also serving to gain familiarity with the technologies used in the project.

A8: Vertical Prototype

A8: Vertical Prototype

- The A8 artifact:
  - corresponds to the implementation of the high priority user stories (features with *);
  - is used to validate the architecture presented;
  - also serves to gain familiarity with the technologies used in the project.
- It must:
  - be based on the LBAW Framework and
  - include work on all layers of the architecture: user interface, business logic and data access.

- The LBAW Framework (template-laravel) includes an authentication system that must be adapted by each group.
- The user stories must include at least a form, an action, an AJAX request, search, and update to the database.
- **See the checklist for what is expected!**

A8: Vertical Prototype

- The vertical prototype must be based on the LBAW Framework.

- PostgreSQL for data persistence.
- Laravel for server-side development.
- HTML, CSS and JavaScript for frontend development.
- Docker for deployment of the product as a Docker container.

LBAW Computational Setup

PostgreSQL

- ➔ PostgreSQL is the relational database management system adopted.
- ➔ Groups have a production database at db.fe.up.pt.
- ➔ For development, a local PostgreSQL can be setup using Docker.

- ➔ SQL is managed using the group's GitLab repository.
- ➔ Do not create or alter the database using the graphical interface.

- ➔ Both the Vertical Prototype and the Product must work on the db.fe.up.pt.

Deploying the Project

- ➔ Docker is used to deploy the application, both for
    - ➔ the Vertical Prototype (A8)
    - ➔ the final Product (A9)

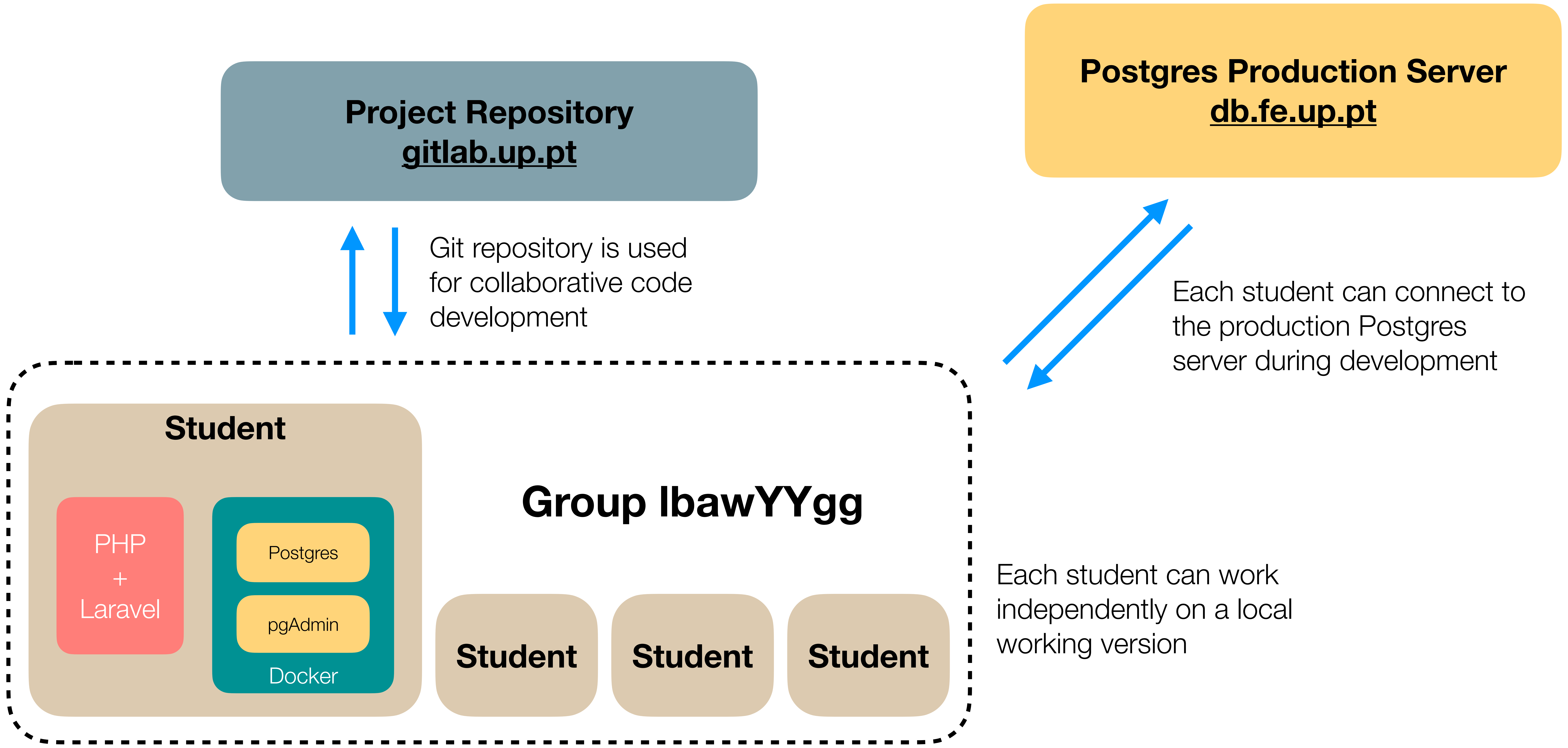
- ➔ Docker is used for
    - ➔ Local setup of PostgreSQL and pgAdmin
    - ➔ Publishing your application at the group's Gitlab Container Registry

- ➔ Each group builds a Docker image of the application and publishes using Gitlab's Container Registry.
    - ➔ This image uses db.fe.up.pt by default.
    - ➔ It can be downloaded to locally view your application.
- ➔ Section "Publishing your image" in templare-laravel guide.

Laravel for Server-Side Web Development

- ➔ The group develops a web application using the Laravel server framework.
- ➔ The group starts the project by copying the files available at the template repository [gitlab.up.pt/lbaw/template-laravel] to their own repository, and then follow the instructions detailed in the README to set up the development requirements.
- ➔ The project uses the PostgreSQL database management system. A local instance is created using the bundled Docker compose file (see the README for the required steps to start it).
- ➔ The group should keep an up to date Docker image of their project in the group's Gitlab Container Registry (see the template README for the required steps).
- ➔ There is a practical guide available to help you build your prototype using the work done in the previous artifacts, namely A3, A6, and A7, see LBAW A8: Putting it all together.

Computational Setup for Development



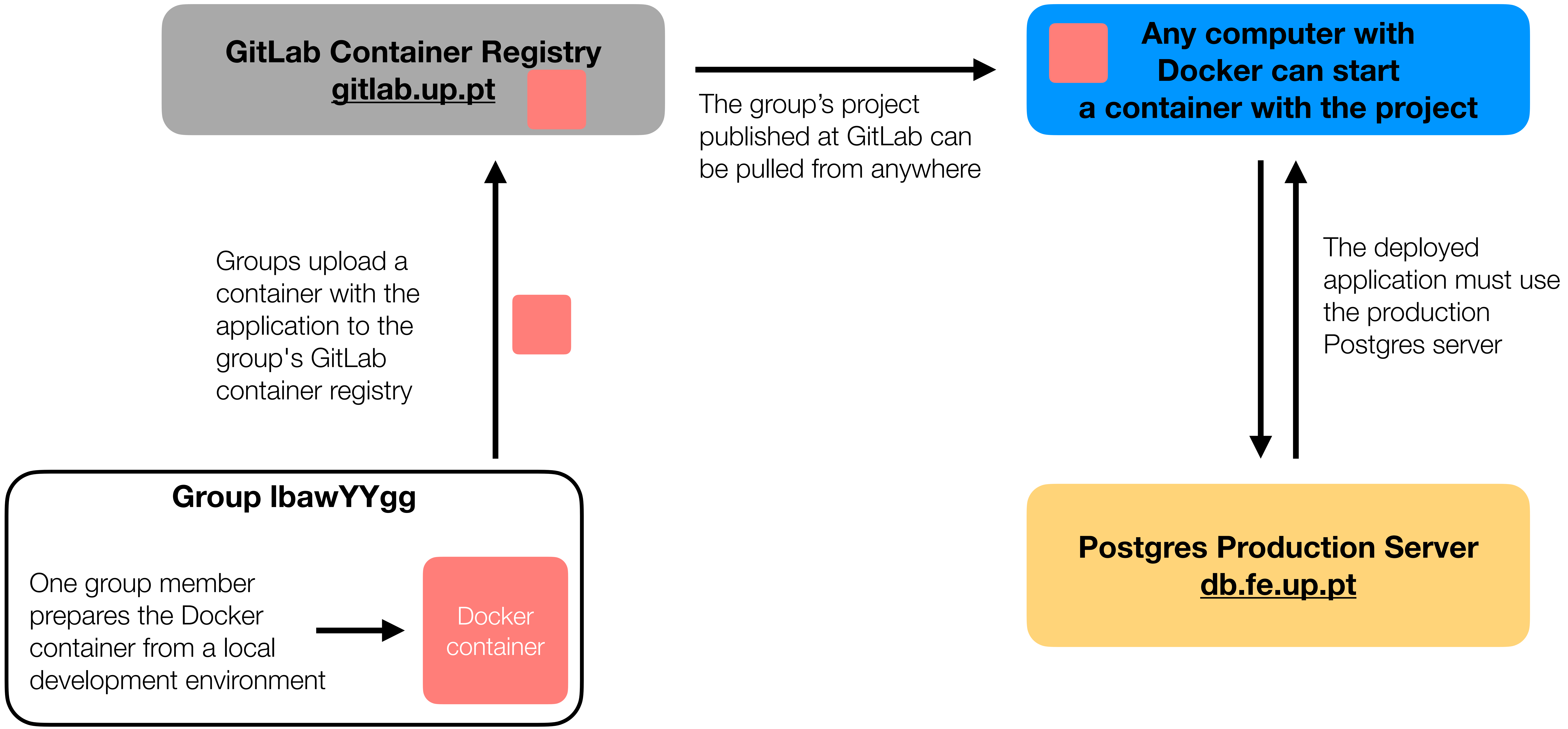
Git repository is used for collaborative code development

Each student can connect to the production Postgres server during development

PHP and Laravel are installed locally, as part of the student setup

Each student can work independently on a local working version

Computational Setup for Deployment


GitLab Container Registry
gitlab.up.pt
The group's project published at GitLab can be pulled from anywhere
Any computer with Docker can start a container with the project
Groups upload a container with the application to the group's GitLab container registry
The deployed application must use the production Postgres server
Group lbawYYgg
One group member prepares the Docker container from a local development environment
Docker container
Postgres Production Server
db.fe.up.pt

Summary

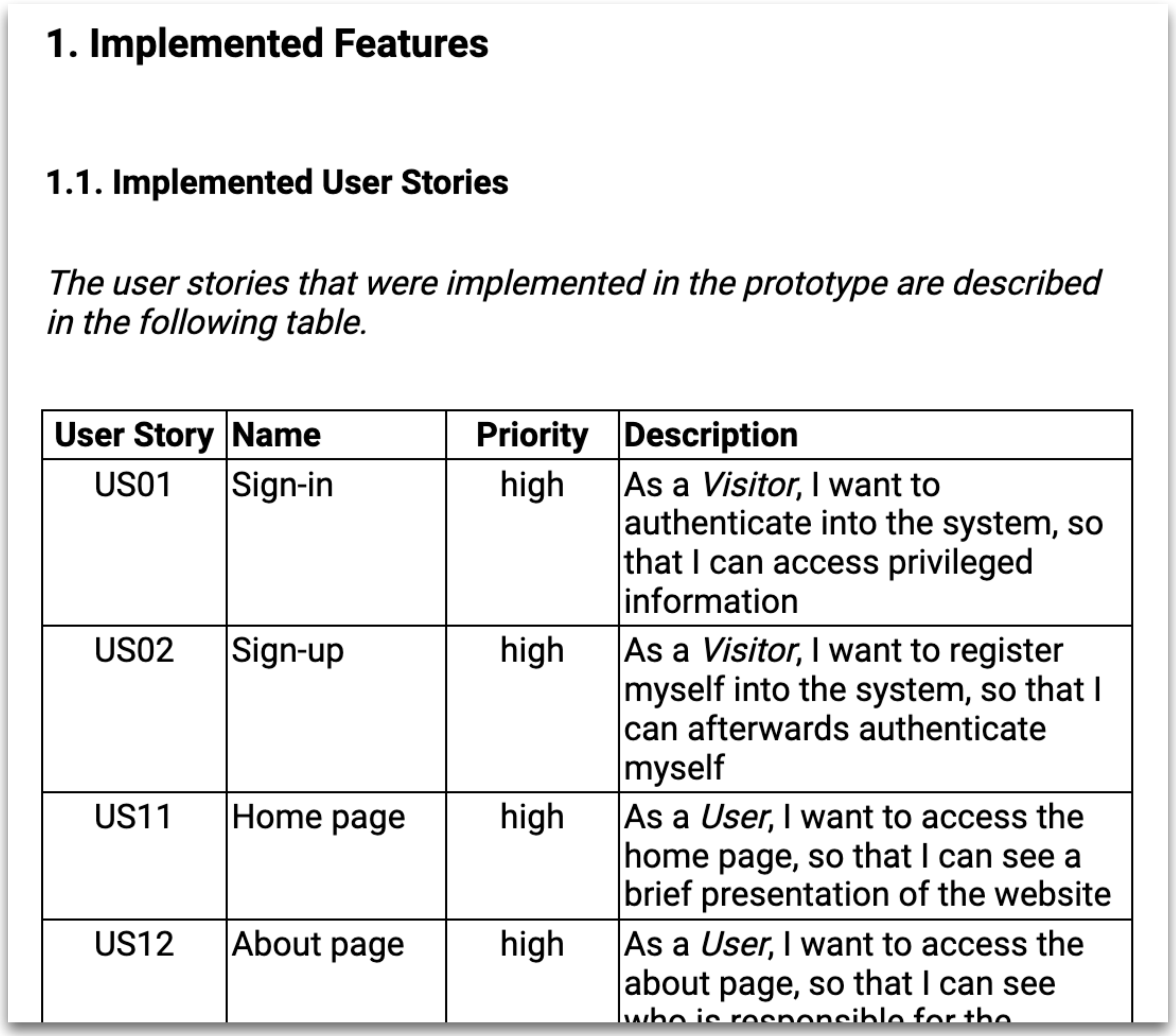
- ➔ Each student has a local Postgres (Docker) and Laravel (local) installation.
- ➔ A GitLab repository is used for code development.
- ➔ Database deployment is done on the production server at db.fe.up.pt.
- ➔ Web application deployment is done
    - ➔ Preparing a Docker container with the web application;
    - ➔ Uploading the container to a group's GitLab Container Registry;
    - ➔ Testing the container pulling it from the Container Registry.

Resources

- A repository with a Laravel setup and instructions to use as a starting point
  - https://gitlab.up.pt/lbaw/template-laravel
  - This repository includes a demo application — Thingy!
- A practical guide to help you build your prototype using the work done in the previous artifacts (A3, A6, A7),
  see "LBAW A8: Putting it all together".
- A8 MediaLibrary example.
- A8 Checklist.

MediaLibrary Example

A8 MediaLibrary - Implemented Features



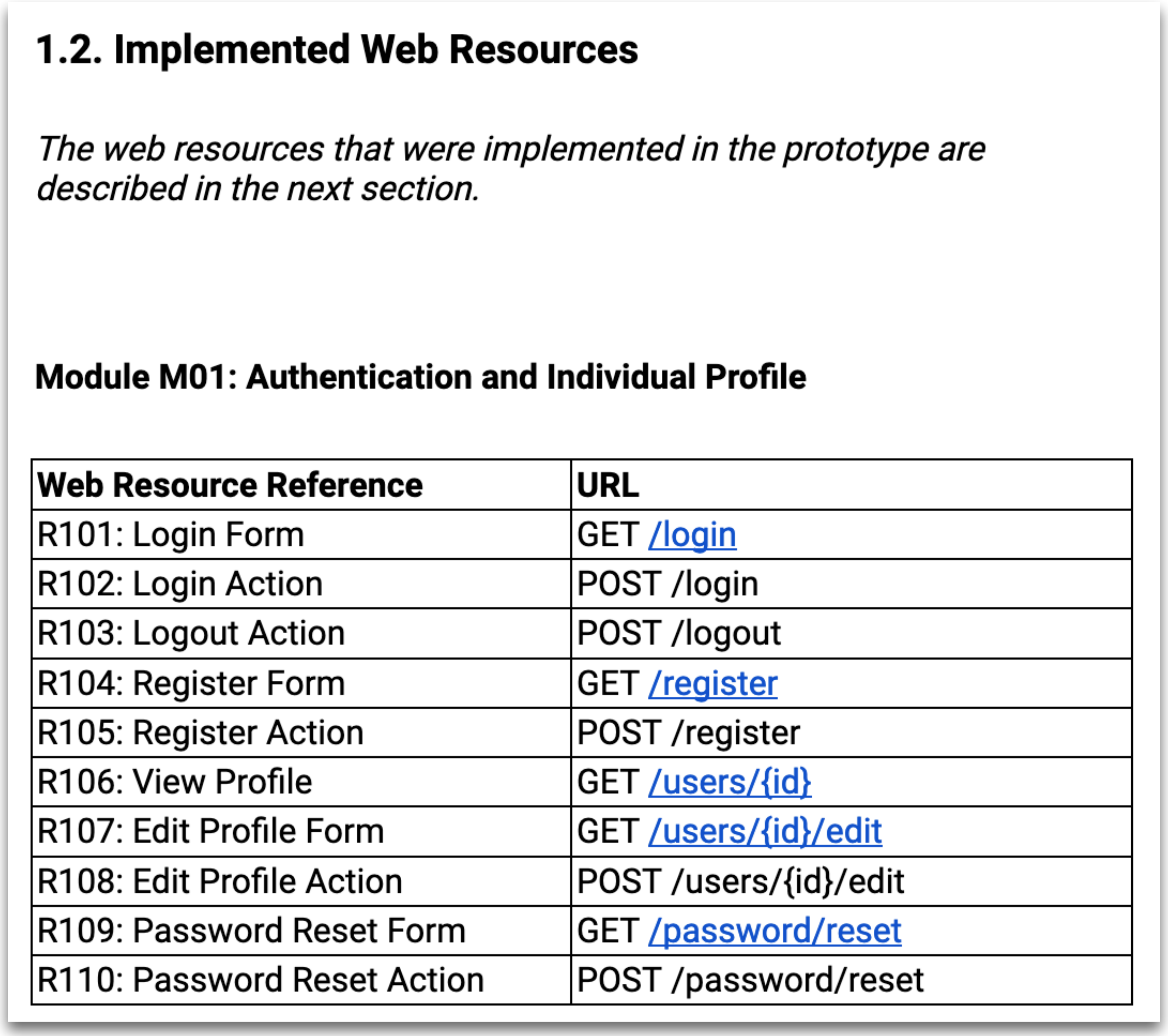
1. Implemented Features

1.1. Implemented User Stories

The user stories that were implemented in the prototype are described in the following table.

User Story	Name	Priority	Description
US01	Sign-in	high	As a *Visitor*, I want to authenticate into the system, so that I can access privileged information
US02	Sign-up	high	As a *Visitor*, I want to register myself into the system, so that I can afterwards authenticate myself
US11	Home page	high	As a *User*, I want to access the home page, so that I can see a brief presentation of the website
US12	About page	high	As a *User*, I want to access the about page, so that I can see who is responsible for the

A8 MediaLibrary - Implemented Web Resources



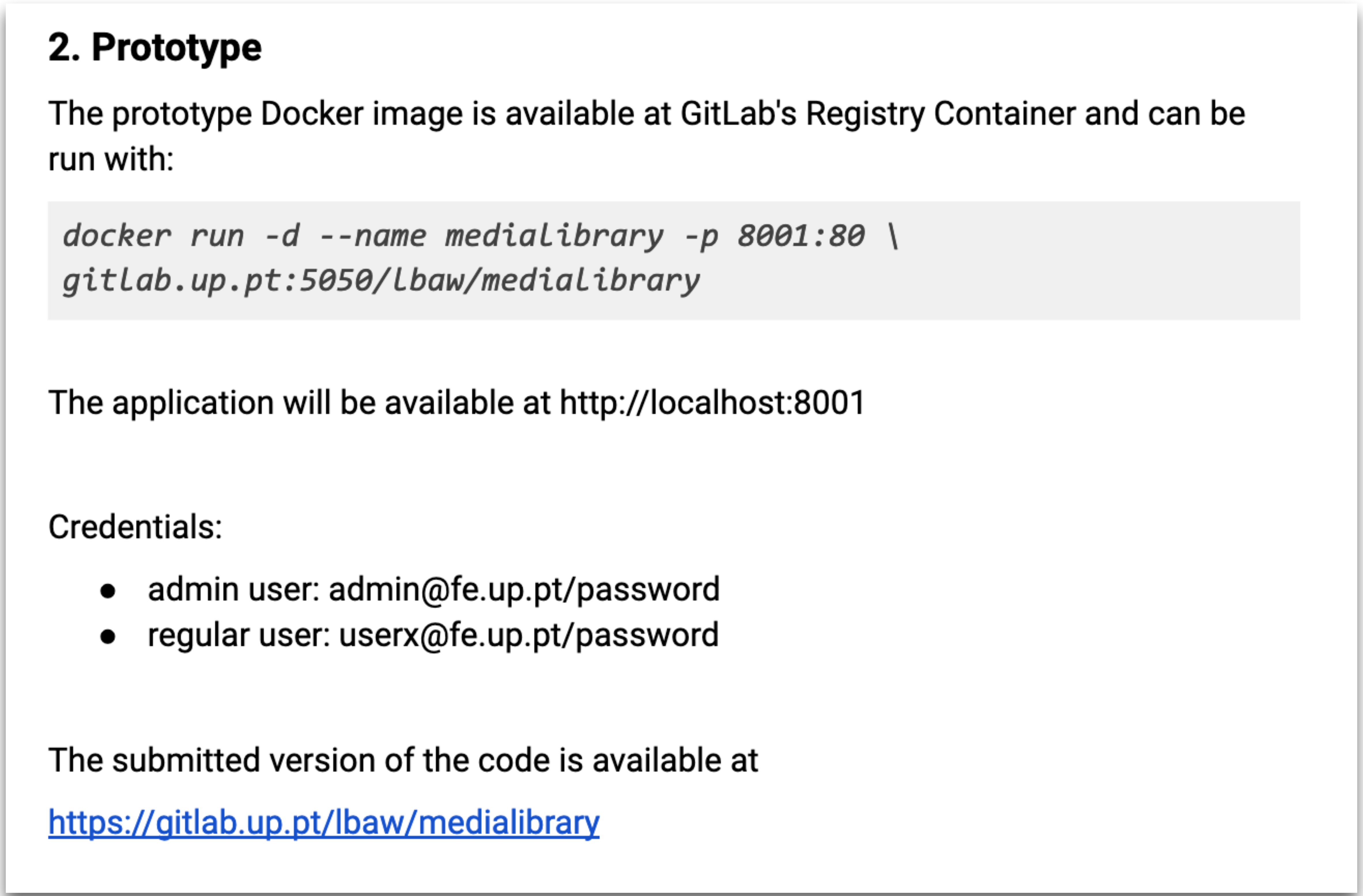
1.2. Implemented Web Resources

The web resources that were implemented in the prototype are described in the next section.

Module M01: Authentication and Individual Profile

Web Resource Reference	**URL**
R101: Login Form	GET /login
R102: Login Action	POST /login
R103: Logout Action	POST /logout
R104: Register Form	GET /register
R105: Register Action	POST /register
R106: View Profile	GET /users/{id}
R107: Edit Profile Form	GET /users/{id}/edit
R108: Edit Profile Action	POST /users/{id}/edit
R109: Password Reset Form	GET /password/reset
R110: Password Reset Action	POST /password/reset

A8 MediaLibrary - Prototype



2. Prototype

The prototype Docker image is available at GitLab's Registry Container and can be run with:

```
docker run -d --name medialibrary -p 8001:80 \
gitlab.up.pt:5050/lbaw/medialibrary
```

The application will be available at http://localhost:8001

Credentials:

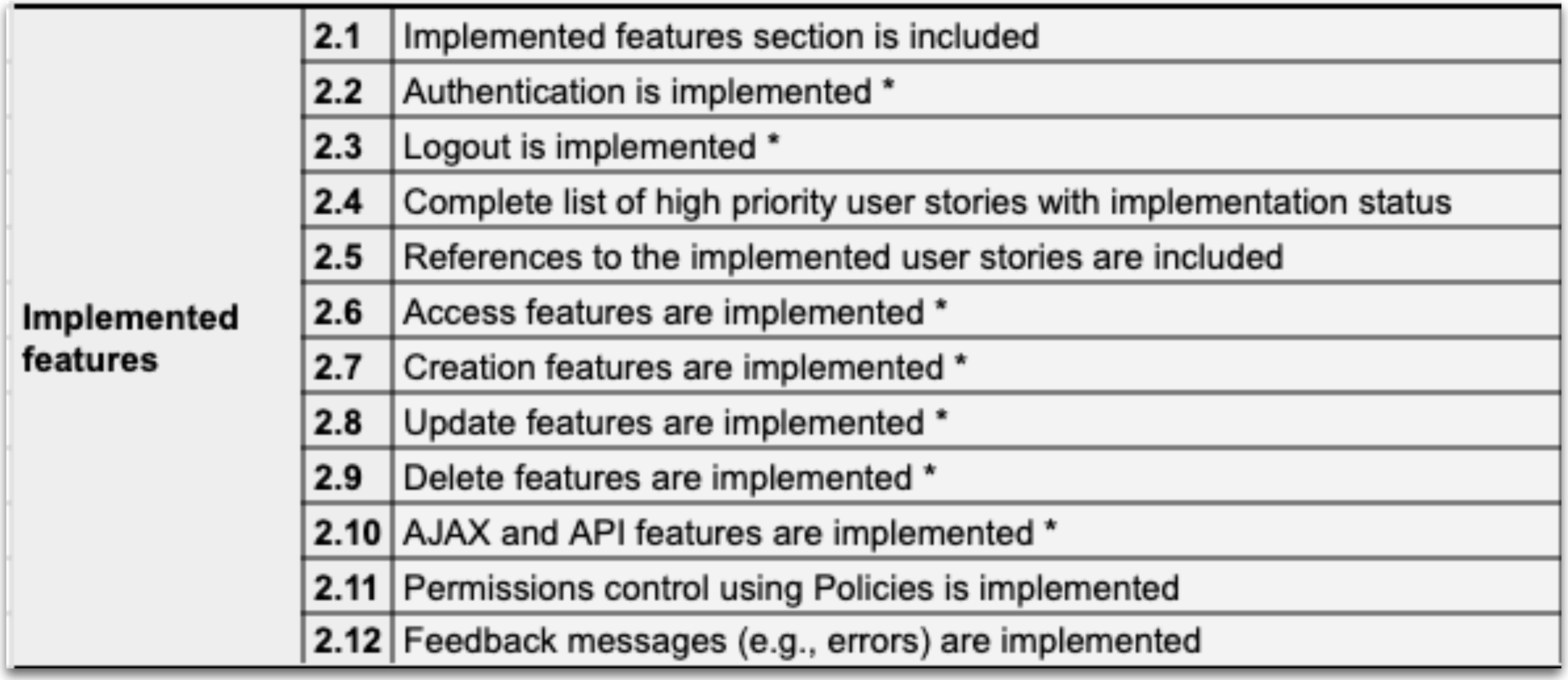
- admin user: admin@fe.up.pt/password
- regular user: userx@fe.up.pt/password

The submitted version of the code is available at

https://gitlab.up.pt/lbaw/medialibrary

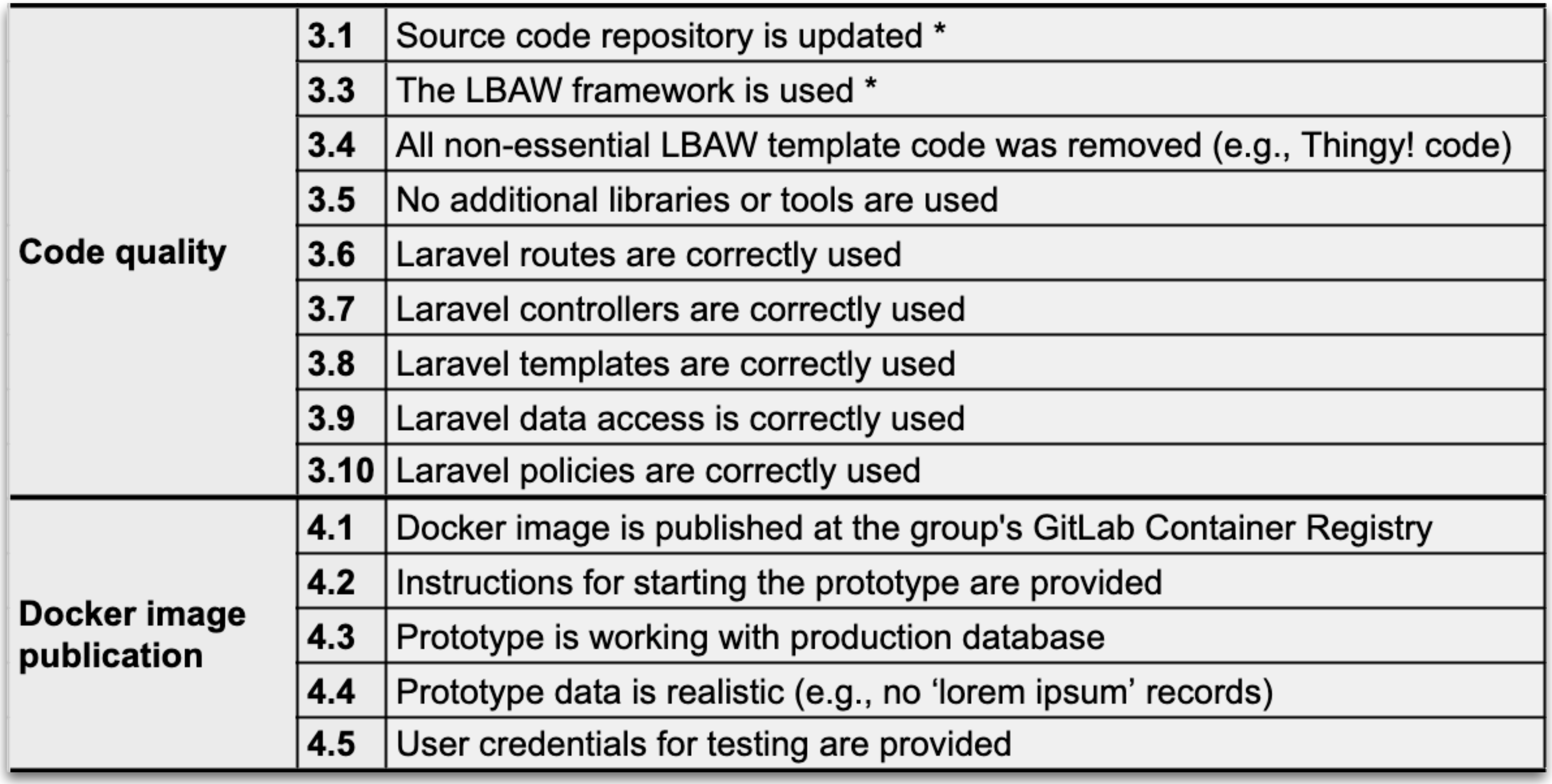
A8 Checklist

A8 Checklist - Implemented Features



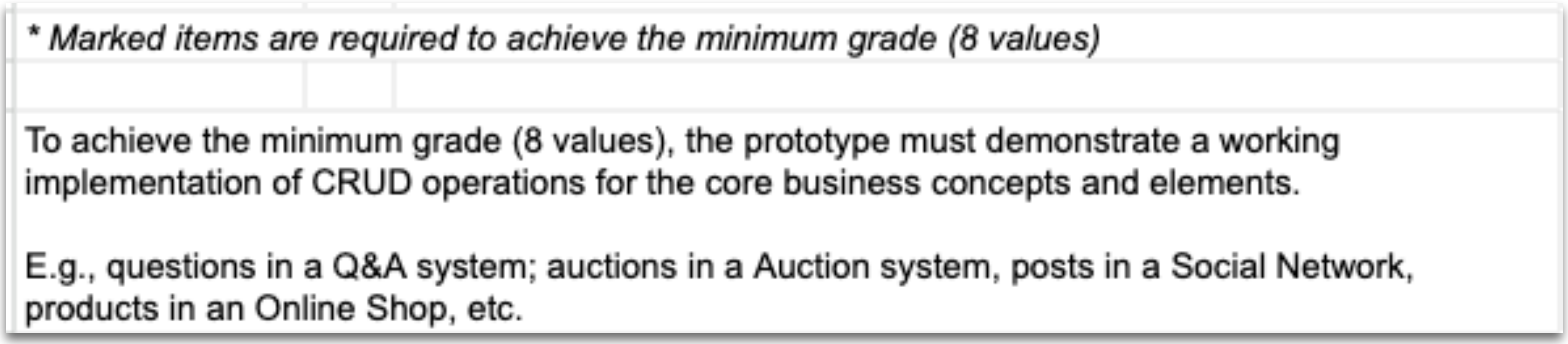
Implemented features	**2.1**	Implemented features section is included
	2.2	Authentication is implemented *
	2.3	Logout is implemented *
	2.4	Complete list of high priority user stories with implementation status
	2.5	References to the implemented user stories are included
	2.6	Access features are implemented *
	2.7	Creation features are implemented *
	2.8	Update features are implemented *
	2.9	Delete features are implemented *
	2.10	AJAX and API features are implemented *
	2.11	Permissions control using Policies is implemented
	2.12	Feedback messages (e.g., errors) are implemented

A8 Checklist - Code and Prototype



Code quality	**3.1**	Source code repository is updated *
	3.3	The LBAW framework is used *
	3.4	All non-essential LBAW template code was removed (e.g., Thingy! code)
	3.5	No additional libraries or tools are used
	3.6	Laravel routes are correctly used
	3.7	Laravel controllers are correctly used
	3.8	Laravel templates are correctly used
	3.9	Laravel data access is correctly used
	3.10	Laravel policies are correctly used
Docker image publication	**4.1**	Docker image is published at the group's GitLab Container Registry
	4.2	Instructions for starting the prototype are provided
	4.3	Prototype is working with production database
	4.4	Prototype data is realistic (e.g., no 'lorem ipsum' records)
	4.5	User credentials for testing are provided

A8 Checklist - Required Minimum



** Marked items are required to achieve the minimum grade (8 values)*

To achieve the minimum grade (8 values), the prototype must demonstrate a working implementation of CRUD operations for the core business concepts and elements.

E.g., questions in a Q&A system; auctions in a Auction system, posts in a Social Network, products in an Online Shop, etc.