



# COMMANGO



**Ahmad Rashad Rafa** <raah191s@student.ju.se>

A Project Work in *Web Development Fundamentals*

Jönköping University 2023

## Table of Contents

Introduction	3
Architecture	4
Database	5
Graphical User Interface	6
Web Application	7

## Introduction

Commango is a website designed to make managing companies easier. With its user-friendly interface, managers can easily add people, tasks, and products to their company's system. It also includes a frequently asked questions (F&Q) section to help users navigate the platform and find the information they need quickly and efficiently. Commango is the perfect solution for streamlining the management process and keeping your company organized. The website results in keeping everything organized and helps the manager save time and avoid errors. With Commango, managers can ensure that their company is running smoothly and efficiently.

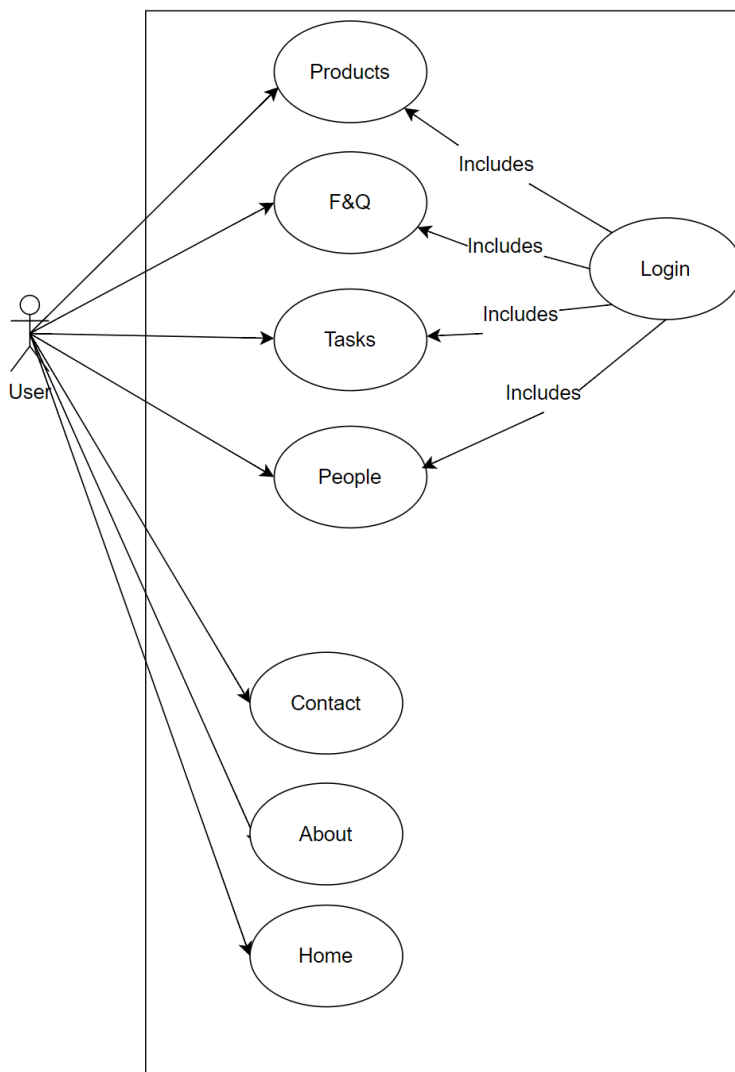


Figure 1: Use case diagram

## Architecture

There are three parts to Commango: a web application, a database, and web browsers. Web application accepts the request and queries the database for the necessary material. Client web browser sends http requests to fetch the website content. If everything went according to plan (success data fetching). The user receives an HTTP response providing the data they requested.

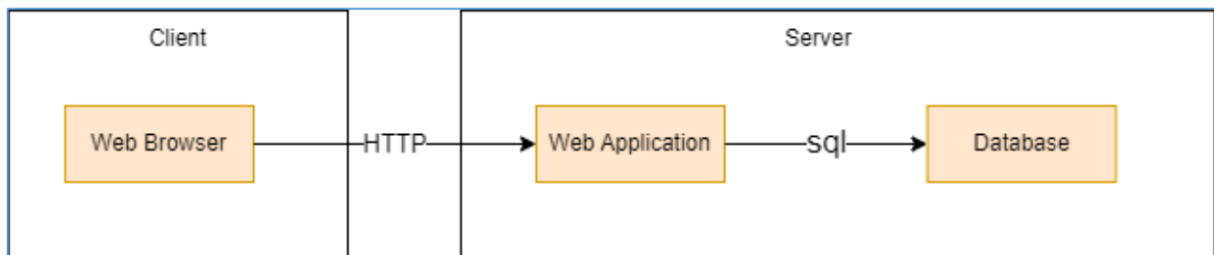


Figure 2: Web application architecture.

## Database

The resources on the website that are stored in a database are the following:

- People
- Tasks
- Products
- F&Q

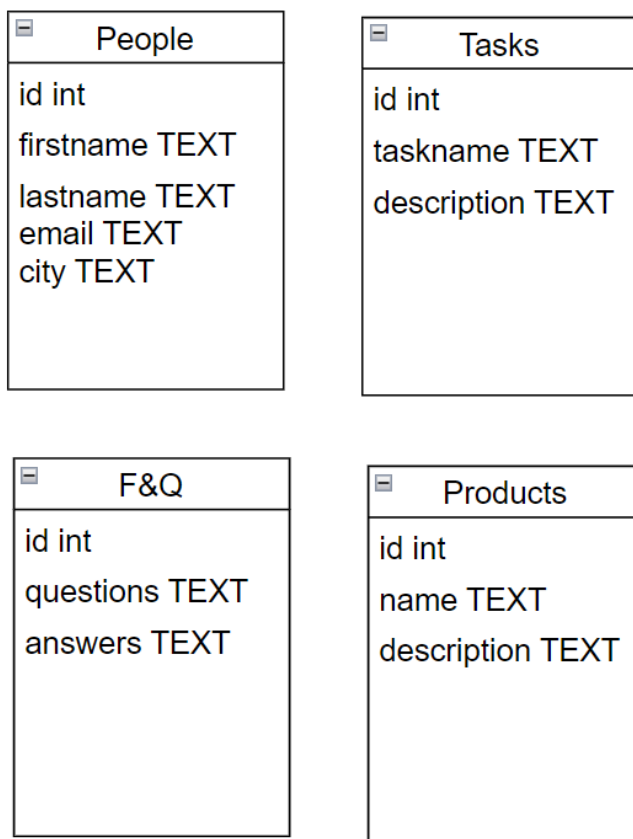


Figure 3: Database Entity Relation Diagram

## Graphical User Interface

The Graphical User Interface (GUI) of the system is built using the Bootstrap framework. Its functionality is tailored to the user's type. Once the user logs in successfully, they will have the ability to perform CRUD (create, read, update, delete) operations on four specific resources within the system (People, Products, Tasks & F&Q).

Start People Tasks Products F&Q Contact About

Welcome to commango  
— Managing your company *has never been easier*

## Create Person

Firstname

Lastname

E-mail

City

Create peoples page

© 2022, Made By Rashad Rafa

Figure 4: GUI Homepage

[Start](#) [People](#) [Tasks](#) [Products](#) [F&Q](#) [Contact](#) [About](#)

# Create Person

Firstname

Lastname

E-mail

City

Create

Peoples page

---

© 2022, Made By Rashad Rafa

Figure 5: GUI Peoples page

[Start](#) [People](#) [Tasks](#) [Products](#) [F&Q](#) [Contact](#) [About](#)

# Create Task

Taskname

Select task stats

Started ▼

Assigned-User

Create

Tasks page

---

© 2022, Made By Rashad Rafa

Figure 6: Gui Tasks page.



[Start](#) [People](#) [Tasks](#) [Products](#) [F&Q](#) [Contact](#) [About](#)

# Create Product

Name

Description

Create

---

© 2022, Made By Rashad Rafa

Figure 7: GUI Products Page.

Start People Tasks Products F&Q Contact About

# Create Question

Question

Answer

Create

Figure 8 Gui F&Q page.

Start People Tasks Products F&Q Contact About Log Out

## Contact us

Full Name

Email Address

Your Message

Submit Form

Figure 9: GUI contact page.

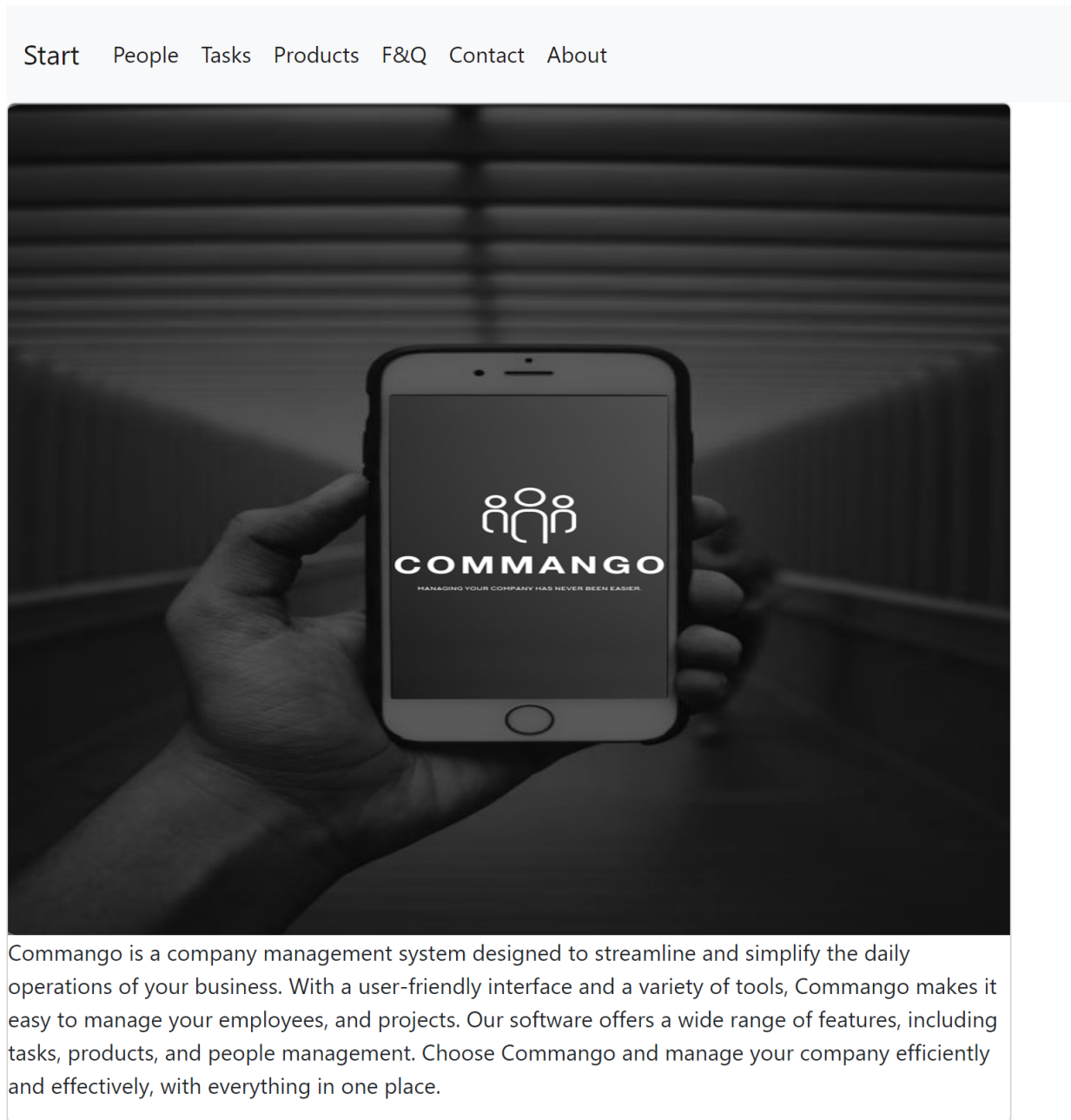
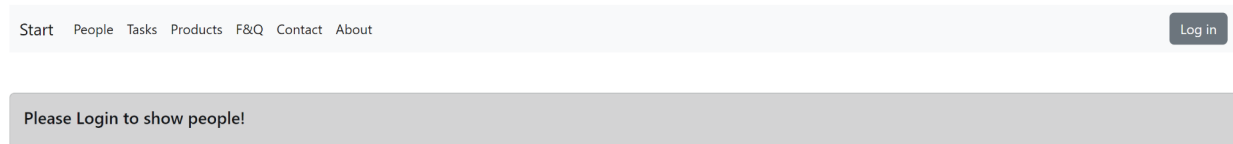


Figure 10 GUI About page.



*Figure 11: Not logged in error message*

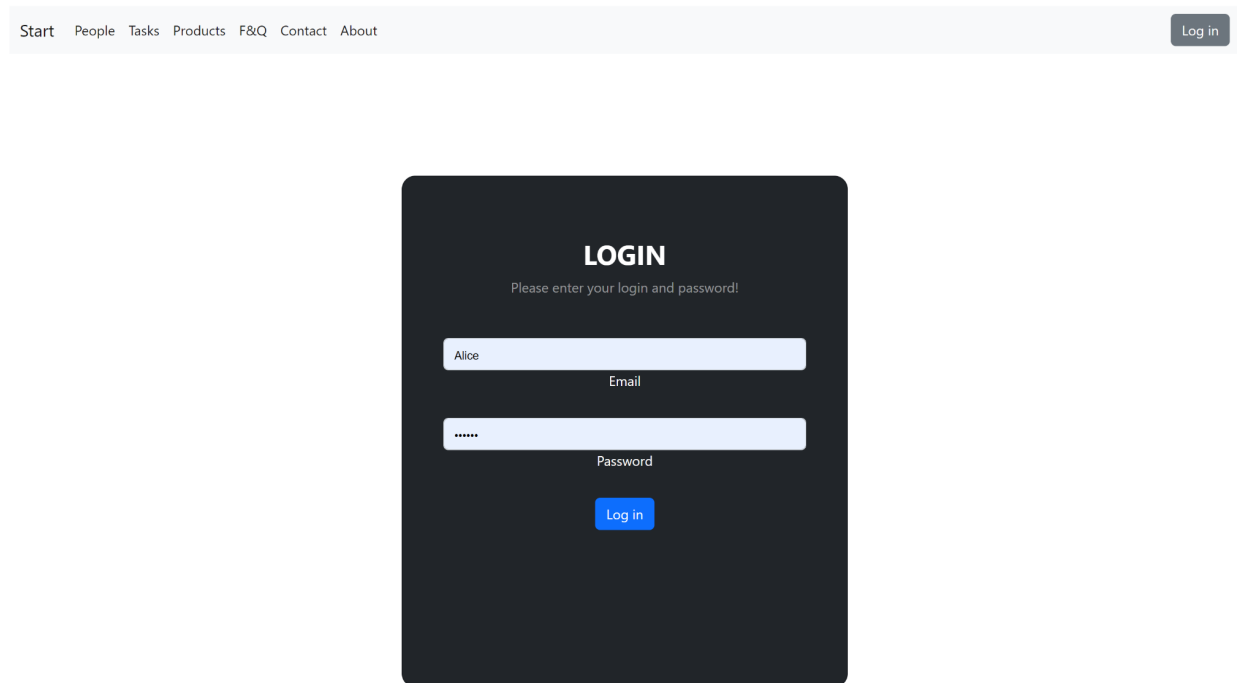


Figure 12: Login Page

## Web Application

Commango utilizes a number of technologies and frameworks to provide its functionality. On the back-end, it uses the Express Framework, Express-Handlebars, body-Parser, Express-Session, SQLite3, and Bcrypt JS. The front-end is built using the Bootstrap framework. Express Framework's middleware is used to handle HTTP requests from the website, and Express-Handlebars is used to save time and not overwrite code for the layouts. The system has a login feature for the administrator, which is handled by Bcrypt JS, Express-Session, and SQLite3. The administrator will have the ability to perform CRUD operations on all implemented resources. Express-Session is used to manage the admin's session when logged in. The project's implementation is divided into different parts. The Database folder contains the database implementation, the Validators folder contains JavaScript code for validating user input, and the Router folder contains the code for the get/post methods. The Static folder contains CSS code that is not part of the Bootstrap framework and images used on the website. The Views folder consists of the Handlebars files, with a Layouts folder where the main.hbs file is implemented, and it contains the website's default structure. The project's entry point is the app.js file, which acts as the connector and has some functions that handle the get/post methods.

```
1  const express = require("express");
2  const expressHandlebars = require("express-handlebars");
3  const expressSession = require("express-session");
4  const bodyParser = require("body-parser");
5  const productsRouter = require("./routers/product-router");
6  const SQLiteStore = require("connect-sqlite3")(expressSession);
7  const authRouter = require("./routers/auth-router");
8  const humansRouter = require("./routers/humansRouter");
9  const tasksRouter = require("./routers/tasksRouter");
10 const questionsRouter = require("./routers/questions-router");
11 const app = express();
12
```

Figure 13: Project packages.

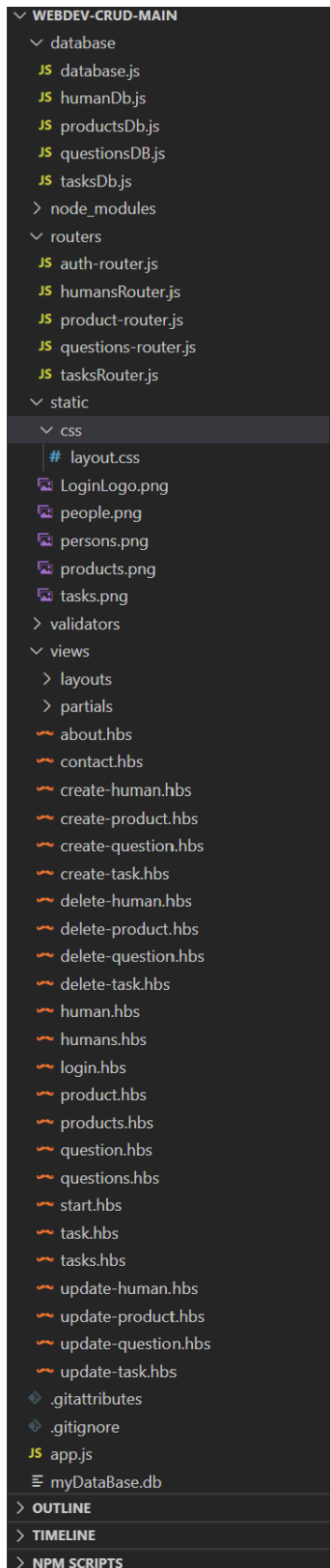


Figure 14: All files.

## Security

With Express-Handlebars, cross-site scripting is prevented. We can compare the entered password to the admin password and hash passwords before saving them thanks to the Bcrypt JS library. Hashing's irreversibility, which offers greater security for accounts, is a big advantage. A hashed password cannot be seen by anyone. The password is kept in the database, though, so even a hacker who gains access to the database won't be able to read it. SQL injections are protected from using placeholders.