|  |
| --- |
|  |
| BTWBeheer  Architecture Document – Individual project |
| |  |  |  | | --- | --- | --- | | Nies,Midas M.F.A. | Demand based - Semester 3 | Software engineering HBO ICT Fontys | |

Contents

[Project Description 2](#_Toc146880252)

[Technology Stack 2](#_Toc146880253)

[Front-End 2](#_Toc146880254)

[Back-End 2](#_Toc146880255)

[Database 2](#_Toc146880256)

[ORM (Object-Relational Mapping): 2](#_Toc146880257)

[Authentication and Authorization 2](#_Toc146880258)

[Deployment 2](#_Toc146880259)

[Project Architecture 2](#_Toc146880260)

# Project Description

BTWBeheer is a web-based bookkeeping application designed for small entrepreneurs and businesses to manage their finances efficiently. Inspired by the success of digiBTW, our goal is to create a user-friendly and feature-rich platform that simplifies bookkeeping tasks, helps users stay tax-compliant, and provides valuable insights into their financial health.

# Technology Stack

## Front-End

* React

## Back-End

* Node.js
* Express

## Database

* MySQL

## ORM (Object-Relational Mapping):

* Sequelize

## Authentication and Authorization

* Passport.js

## Deployment

* Hosting on cloud.
* Docker for containerization (optional).

# Architecture

## Server projectA screenshot of a computer Description automatically generated

For my file structure I am using Express common practices as you can see below:

Models: These models represent the structure of the bookkeeping data, such as accounts, invoices, etc. I am using an Object-Relational Mapping (ORM) library: Sequelize.

Controllers: Controllers handle the logic for API routes. Each controller should correspond to a specific resource or functionality, like creating a transaction, fetching account balances, or generating reports.

Routes: Routes should map to controller methods and follow RESTful principles. For example, routes like /api/invoices, /api/relations, etc.

Middleware: I use middleware functions for tasks like authentication, logging, error handling, and request validation.

Authentication: Passport.js and Bcrypt, for password hashing and JWT tokens.

## Communication

My React components will make HTTP requests to my Express API endpoints to perform CRUD (Create, Read, Update, Delete) operations on bookkeeping data. The API will be MVC and RESTful which will make communication between front and backend easy.

## Version control

For my version control I will use GitHub which will also initiate automated unit tests. I will do feature-based branching when the project it initialized, to make sure everything is ordered and will not be a big mess.

## Automated deployment and testing

I will automate deployment and testing via GitHub actions, When I push to main or merge: GitHub will first run my unit tests, if they are all valid it will continue to create 2 docker images, one for the frontend, one for the backend. The docker images will be pushed to my docker account which will be linked to azure to automatically deploy the real application.

# C4 Diagrams

## C1

A diagram of a customer service

Description automatically generated

## C2

A diagram of a software application

Description automatically generated

## C3

A diagram of a web application

Description automatically generated