VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Duy Le

WEB APPLICATION DEVELOPMENT

with React, Node.js and SQL

Technology and Communication
2024

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Duy Le
Title	Web Application Development with React, Node.js and SQL
Year	2024
Language	English
Pages	38
Name of Supervisor	Anna-Kaisa Saari

The primary goal of the thesis was to create a full-stack web application that combines front-end and back-end technologies to optimize tech operations which shows the users how the application looks like and supports them to use its functions. The web application uses several different programming languages and software frameworks to provide a stable platform. This thesis outlines the basic advantages and disadvantages of a web application as well as suggests solutions for these problems.

The outcome of this thesis is an example of building and operating a basic web app development. Users can operate it by themselves or develop it on a large scale. Besides, they can improve the basic knowledge and important features of technological web applications, which might be necessary in their programming development career.

The project was made on the full-stack web application structure. The application utilizes the technologies Node.js, Express.js and Sequelize for the back end, and React, Axios and Yup for the front-end. The database management system of the program is MySQL.

Keywords React, Node.js, SQL, full-stack, front-end, back-end

CONTENTS

LIST OF FIGURES AND TABLES

1 INTRODUCTION

1.1 Overview

In the ever-changing world of technology, web application fusion has emerged as a key component of contemporary innovation. The connection of different technologies through web apps has pushed civilization into a new era of efficiency, accessibility, and connectivity as the digital domain continues to develop. This thesis explores the significant effects of incorporating technology into online applications, examining their background and present situations. Moreover, the web application serves as a significant tool for enhancing connectivity in the digital age. With its diverse functionalities and benefits the application plays a important contribution in fostering communication, collaboration, and information exchange in today's interconnected world.

1.2 Expectation

The result of this thesis will be important information on the best methods for developing and putting into practice complete web applications and favorable for those engaged in software development.

In conclusion, the aim of the research is to emphasize the significant effects that web application integration has on users, and innovation. Users may actualize an effective, collaborative, and sustainable future by utilizing web applications to fully utilize the revolutionary power of technology.

2 THE BACK-END SIDE

2.1 Overview

The back end is the part of a web application that is not visible to the user. It refers to the way a website functions and all the components that help to deliver that functionality. Processing requests from the front end, carrying out relevant tasks (such as data retrieval, modification, and storage), and returning the results to the client are its main responsibilities. Numerous programming languages and frameworks are available for use in back-end development. JavaScript is currently widely utilized for front-end and back-end development due to the popularity of Node.js. The server part of this thesis project is made using Node.js. /1/

The work of back-end developers applies to the server side of operations. In back-end development, working with databases to store and retrieve data is common. Depending on the needs of the project, developers must be knowledgeable with a variety of database management systems, such as MySQL, PostgreSQL, and MongoDB. /1/

Application Programming Interfaces, APIs, are frequently created by back-end developers to facilitate communication between front-end and back-end systems. The endpoints and data types that the front-end and back-end may communicate with each other are specified by APIs. All things considered, the back end is essential to full-stack web development since it establishes the framework for online application functionality and data administration.

2.2 Node.js

2.2.1 Introduction of Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment and library that runs web applications outside of the client's browser. It is built on

Google Chrome’s V8 engine, which makes its execution time very fast, and it runs very quickly. The asynchronous, non-blocking I/O paradigm of Node.js is well-known for enabling effective management of several concurrent connections. Building scalable applications that can manage several connections at once is much easier with the asynchronous paradigm of Node.js. Web servers, APIs, and backend services are frequently powered by Node.js. Because of its event-driven architecture, it excels at managing I/O-intensive activities including database transactions, file operations, and network requests. /2/

NPM is a popular Node.js package library. It accesses a huge ecosystem of open-source libraries and tools. NPM allows developers to install, manage, and share packages of reusable code easily, speeding up the development process. Together with NPM, Node.js is a popular choice for developing contemporary online applications and backend services because of its adaptability and ease of use in web applications. /2/

2.2.2 Influence of Node.js to Full-stack Web Development

Node.js has become an excellent option for full-stack web development due to its efficiency, scalability, and the ability to use JavaScript for both frontend and backend development. To be able to run JavaScript code outside of a web browser, runtime environment is needed. Therefore, backend development primarily uses Node.js on the server side. Node.js allows developers to create dependable, and fast online applications.

Node.js is built on an event-driven, non-blocking I/O model. Because of this, Node.js can manage several connections at once without being stopped, which is essential for managing a lot of requests at once in web applications. Consequently, Node.js exhibits exceptional performance and extensibility in managing real-time applications with substantial traffic levels. Among the many characteristics of

Node.js are its considerable library of JavaScript modules, which makes online application development easier, its capacity to manage several client requests at once, and its support for two-way real-time web applications. /2/

Node.js is a great option for creating applications that can be quickly deployed and expanded on the cloud because it is also very compatible with cloud-based architectures. Moreover, Node.js boasts a sizable and vibrant developer community that offers a wealth of information and assistance to anyone utilizing technology. Overall, Node.js influence extends beyond just the technical aspects, shaping the way developer architect and build web applications in the modern era.

2.3 Express.js

2.3.1 Introduction of Express.js

Express.js, usually known as "Express," is a simple, quick, and Node.js backend framework that offers strong functionality and tools for constructing scalable backend applications. It is designed for building web applications and APIs, providing a robust set of features for developing server-side applications. /6/

One of the core features of Express.js is its middleware support, which enables programmers to create modular, reusable components that are capable of handling requests and answers. The duties that middleware functions can carry out include error handling, logging, authentication, and request processing.

Express provides a robust routing system that enables developers to characterize routes for handling various HTTP requests (GET, POST, PUT, DELETE). Routes can be defined for specific URLs and HTTP methods, allowing for clean and organized code. Express.js provides utility methods for working with HTTP requests and responses, making it easier to set headers, cookies, and status codes. Express.js handles the routing and middleware aspect of the process, allowing developers to focus on the critical business logic of these features while designing live collaboration solutions. /3/

2.3.2 Influence of Express.js to Full-stack Web Development

Express.js is popularly used for building web applications, ranging from small personal projects to large-scale enterprise applications. Besides, its impact on full-stack development is significant and multifaceted, influencing both the frontend and backend aspects of web development. /4/

Express.js is a simple framework that makes it easy and quick for developers to construct online applications. Express.js is used widely in the JavaScript/Node.js ecosystem — applications, API endpoints, routing systems, and frameworks can be developed with Express.js. Furthermore, Express.js provides developers with a high degree of flexibility and customization, allowing developers to tailor the framework to suit their specific requirements. /6/

To summarize, Express.js has significantly influenced full-stack web development by streamlining server-side development, enabling effective routing, encouraging code reuse, integrating with frontend frameworks seamlessly, supporting a variety of databases, guaranteeing scalability and performance, creating a thriving community, and providing flexibility and customization. Because of its adaptability and simplicity of use, developers who want to create cutting-edge, reliable, and scalable web applications choose to work with it.

2.4 Sequelize

Sequelize is a promise based ORM for Node.js which incorporates robust reading applications, transaction relationships, support, and loading. It allows connecting relational databases with JavaScript objects while abstracting away the intricacies of database queries and maintenance. Sequelize supports several database languages, including PostgreSQL, MySQL, SQLite, and MSSQL. Sequelize simplify CRUD (Create, Read, Update, and Delete) activities by offering ways for creating, querying, updating, and deleting database records. /5/

3 THE MYSQL DATABASE MANAGEMENT SYSTEM

3.1 Overview

3.1.1 Introduction

MySQL, an open-source Relational Database Management System (RDBMS), allows users to store, manage, and retrieve structured data effectively. MySQL is extensively used for numerous purposes, including small-scale projects, large-scale websites, and enterprise-level solutions. Because MySQL is offered under the GNU General Public License (GPL), anybody can use, modify, and distribute it without restriction. MySQL arranges data into tables with rows and columns and established relationships between them using the relational paradigm. Effective data processing and querying are made possible by this structure. SQL is the main query language that MySQL employs to communicate with the database. SQL gives users the ability to do several tasks, including inserting, deleting, and altering data. /7/

MySQL is known for its fast performance, particularly in read-heavy workloads. It provides effective database operations by utilizing several optimization strategies, including caching, query optimization, and indexing. MySQL is compatible with several operating systems, including Windows, Linux, macOS, and systems that resemble UNIX. Its cross-platform flexibility contributes to its adaptability and widespread use in a variety of settings.

Overall, MySQL continues to be a highly valuable tool for efficiently managing data with strong performance and security. Proficiency in MySQL equips developers with the skills needed to leverage data effectively, empowering them to adapt to the evolving technology and business landscape. /7/

3.1.2 Advantages

Delving into the advantages of MySQL (MySQL Database Management System) can provide valuable insights for developers into MySQL utility and potentiality for various applications. MySQL's free and open source nature makes it the top option for startups and developers. Because of its cross-platform compatibility, researchers may utilize MySQL in a variety of computer settings without having to make substantial changes, ensuring deployment flexibility. /8/

Scalability, high performance, stability, ease of use, cross-platform compatibility, standard SQL support, security features, community support, and integration with well-known tools are just a few of the many benefits MySQL provides to researchers. MySQL is an appealing option for organizing and evaluating research data across several areas and applications because of these benefits. /8/

3.1.3 Disadvantages

MySQL has stability difficulties and is prone to corruption in certain use scenarios. Stability issues can manifest as crashes or unexpected behavior under heavy loads. Database corruption can occur due to reasons of hardware failure or software bugs. While this criticism is rarely voiced in general, there have been widespread concerns concerning data corruption when auditing or making transactions. /8/

Relational databases are meant to store structured data with well-defined relationships. They may be unsuitable for storing unstructured or semi-structured data, such as documents, photos, and multimedia information. While MySQL has a large and active community, official support from Oracle (the current owner of MySQL) may not be as comprehensive or responsive as with other commercial RDBMS vendors. Unlike commercial and paid database vendors that users of MySQL may not receive the same depth of assistance or feature development. This is a negative point to compare with those using Oracle Database or other premium database providers. /10/

3.2 Database Structure

MySQL organizes data into databases, which serve as containers for tables, indexes, views, stored procedures, and other objects. Each database is identified with a unique name and can contain multiple tables.

Table provides a comprehensive set of functionalities for designing, managing, and querying database tables, enabling efficient data storage and retrieval within MySQL database. Tables consist of rows and columns, where row represents one data record or tuple, and a cell stores actual data with a certain data type. Tables are created using the CREATE TABLE statement and can be modified using ALTER TABLE statements. /9/

Rows in MySQL represent individual records or entries in a table. Each row contains data for each column defined in the table.

Columns define the structure of a table, indicating the types of data that can be stored in each field. Integers, dates, texts, floating-point numbers, and binary data are examples of common data types. Each column has a name and data type, and additional constraints can be applied, such as NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, etc.

A primary key uniquely identifies each row in a table. Primary keys are used to ensure data integrity and to provide a way of referencing specific records. /10/

A foreign key establishes a relationship between tables. It refers to the primary key of another table and is used to link related data. Foreign keys are employed to ensure data consistency and enforce referential integrity by connecting records from one table to rows in another. /10/

An index is a set of pointers that is logically ordered by the values of a key. Indexes provide quick access to data and can enforce uniqueness of the key values for the rows in the table. /9/

Constraint is to preserve data consistency and integrity; constraints are guidelines that are applied to the data in a table.

Data Types can store a wide range of data types, including texts, dates, times, integers, floating-point numbers, and more by MySQL. Selecting the proper data type for every column is crucial for effective data storage and retrieval.

A trigger in DBMS is a special type of stored procedure that is automatically executed in response to certain database events such as an INSERT, UPDATE, or DELETE operation. /11/

3.3 Implementation of MySQL in Software Engineering

Web applications often utilize MySQL as their backend database. MySQL is supported as the main database option by frameworks such as Django (Python), Rails (Ruby), and Laravel (PHP). MySQL can support any web application use case. It currently is used in some of the largest web applications, such as eBay, Dropbox, and Twitter. /12/

MySQL is used in software engineering data warehousing solutions, which combine, store, and analyze massive amounts of data for business insight. MySQL can house immense amounts of data to be analyzed. /12/

MySQL is frequently used in the backend systems that enable mobile applications. MySQL is used by mobile applications to store and manage data on the server side for features like content delivery, data synchronization, and user authentication.

MySQL is frequently implemented in the field of software engineering for gaming to manage player data, game state data, leaderboards, and other backend features. MySQL is frequently utilized by the top game production firms, such as Activision Blizzard, as the database back end for numerous video games. /12/

4 THE FRONT-END SIDE

4.1 Overview

The front-end refers to the part of a website or web application that users interact with. Front-end refers to anything that consumers view, touch, click, or interact with on their screens. /1/

Front-end covers components including design, buttons, forms, navigation menus, and multimedia files like pictures and videos. Creating an aesthetically pleasing, intuitive, and user-friendly interface that helps people do activities quickly and joyfully is the main objective of front-end development. Front-end developers construct the front-end for a website or an application using many technologies, languages, and frameworks. Front-end developers not only establish the overall look of a site but also create its interface and user experience. /1/

Front-end developers typically work closely with others—web designers, user experience (UX) analysts and back-end developers—to ensure they meet the specifications. To design cutting-edge user interfaces, front-end developers also need to keep up with the newest developments in web development and technology. Front-end developers ensure the application or website is search engine optimized and accessible to people with impairments. To enable these, a thorough grasp of web standards and best practices is necessary. /1/

4.2 React

React.js is an open-source JavaScript toolkit designed with precision by Facebook to ease the complicated process of designing dynamic user interfaces. It is possible for developers to create modular user interface elements that can be used in many contexts and that automatically update as data changes. It makes it simple to create intricate and dynamic user interfaces. /13/

In React, applications are developed by creating reusable components that can be understood as independent Lego blocks. These components are individual pieces of a final interface, which, when assembled, form the entire user interface of the application. React component-based design makes it easier to maintain applications and reuse code. /13/

React's Virtual DOM is a lightweight version of the Real DOM. Real DOM manipulation is much slower than virtual DOM manipulation. When an object's state changes, the Virtual DOM updates only that object in the real DOM, not all of them. React virtual DOM technology makes component rendering and updating fast, contributing to its excellent performance. /15/

React is an excellent choice for web applications. Its extensibility and adaptability also make it a popular option for developing intricate software systems and applications.

4.2.1 Main Features

React allows users to specify how the UI appears depending on the application state. This makes it more comfortable to understand and debug code even if there are many errors occurring. React is heavily component based. When understanding it maintaining the code when working on larger scale projects is easier. React updates the user interface quickly by using a virtual DOM. React updates the virtual DOM whenever a component's state changes, then determines the most effective method to update the actual DOM. /14/

JSX is JavaScript syntax extension. It is not necessary to use JSX in React development, but it is recommended. Using JSX makes it easier to create React components. /14/

React uses one-way data flow, making it simple to comprehend the application. Unidirectional data flow enables to maintain predictable behavior and simplifies debugging. /14/

There is a huge and active developer community that uses React, so documentation, tutorials, and support are easily accessible. This simplifies the learning and troubleshooting process for developers using React. The community continuously provides new libraries and tools to improve the development experience.

4.2.2 Key Concepts

React application consists of components and elements. Elements are lightweight, immutable objects that represent the UI components users want to render on the screen. Elements are typically created using JSX.

Components are the building blocks that comprise a React application representing a part of the user interface. /15/

In React, understanding the concepts of props and state is fundamental to building dynamic and interactive components. Props transmit data and event handlers to a component's children (the children refer to the nested elements or components that are passed as content to another component). Props are immutable and cannot be modified by the child component. State is used to handle data that changes over time within a component. Unlike props, state is changeable and may be modified with the `setState()` function. /15/

React provides a collection of life cycle events (or callback API) to attach functionality (the process of connecting or associating specific methods or behaviors with different stages or events in a React component's lifecycle). These methods are executed during the various stages of the component. The components stages are the different phases that React component goes through during its lifecycle: initialization, mounting, updating, and unmounting. /14/

4.2.3 Advantages

React uses Virtual DOM concept to check and update the HTML document. Virtual DOM is a special DOM created by React. Virtual DOM represents the real DOM of the current document. Whenever there is a change in the document, React checks the updated virtual DOM with the previous state of the Virtual DOM and update only the different in the actual/real DOM. This improves the performance of the rendering of the HTML document. /14/

In addition, components encourage code reuse, making it easier to maintain and expand programs. React also has a huge and active community, as well as several libraries, tools, and resources to assist developers in creating more efficient applications. React also supports mobile development with React Native, a React-based framework. /15/

4.2.4 Disadvantages

React.js is a library for creating user interfaces. To construct complicated applications, developers must rely on other libraries or frameworks, such as Redux for state management. This might result in greater complexity and boilerplate code in larger projects. /16/

React has introduced JSX to operate with HTML and JavaScript. JSX is basically JavaScript coupled with HTML syntax. It allows combining HTML and JavaScript, but introduces several new attributes and syntaxes, making it challenging to deal with first starting development with React. /16/

4.3 Axios

Axios is a popular JavaScript library for sending HTTP requests via the browser or Node.js. It provides a simple API for developers to send asynchronous HTTP queries to servers and manage results.

Axios streamlines the process of sending HTTP requests by offering a clear and intuitive API. It is promise-based; thus, developers may create asynchronous code with `async/await` or using `.then ()` syntax. /18/

Axios may be used in either browser-based or server-side applications. On the server-side it uses the native node.js `http` module, while on the client (browser) it uses `XMLHttpRequests`. /17/

Axios has robust error handling features, allowing developers to gracefully handle a variety of HTTP problems. This involves managing network problems, timeouts, and status code issues.

4.4 Yup Schema Builder

Yup is a JavaScript schema builder that handles value parsing and validation. Form validation is frequently handled in React.js apps using form frameworks like Formik. With Yup, the developer can define a schema (or structure) of the expected data specifying its data type and whether it is required or not. /19/

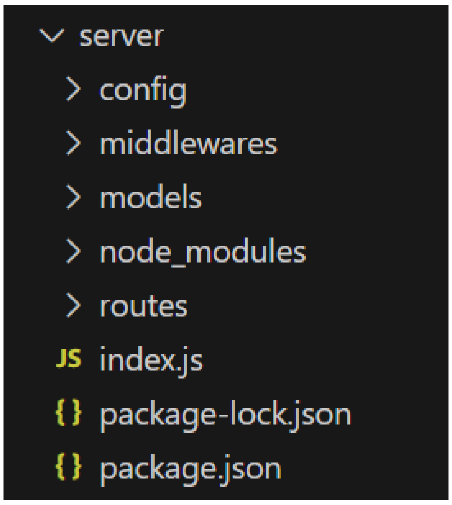
Yup offers a powerful and flexible solution for defining and validating data schemas in JavaScript applications, particularly in the context of form validation. Yup simplifies the process of enforcing data integrity and ensuring that the application handles user input correctly.

5 WEB APPLICATION IMPLEMENTATION

5.1 Overview

The main purpose of this thesis is to create a web application with basic functions to consolidate information and knowledge for those who want to start web development projects. The core features of the web application are representing machines and user's metrics, allowing users to modify machines description data, and supporting users on authentications. The project used very simple technologies to make and took full advantage of these tools to create a complete web application. Besides, the thesis also mentions and emphasizes important items of the above technologies for readers to easily select.

5.2 Back-end Implementation

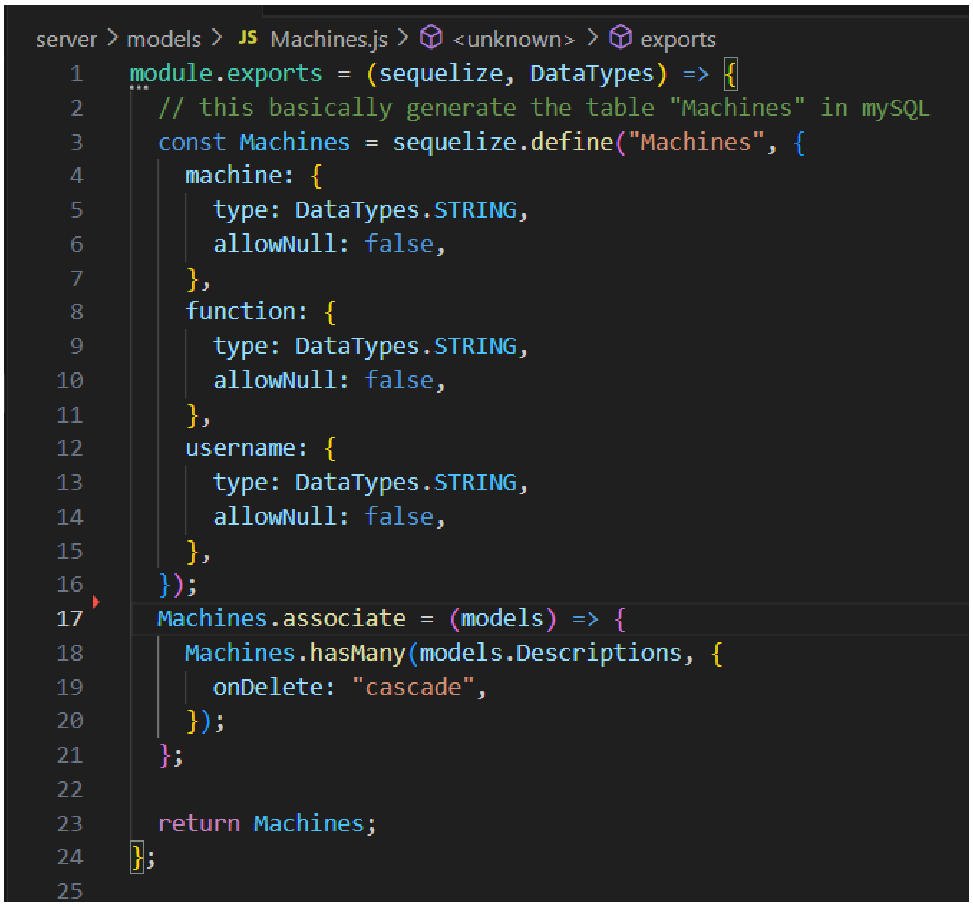


```
∨ server
  > config
  > middlewares
  > models
  > node_modules
  > routes
  JS index.js
  {} package-lock.json
  {} package.json
```

Figure 1. Back-end directory

Figure 1 presents the server directory that contains the back-end source code and the configuration of the database storing by MySQL DBMS. The `config` folder included config JSON file that provides different set of database connections settings for different environments. The `middlewares` folder contained `AuthMiddleware.js` file for a middleware function used for authenticating requests by verifying JWTs passed in the request headers. The `models` folder essentially sets up the code with database configuration provided, loads all the models defined in it, establishes associations between them, and exports the database object to use throughout the application. The `node_modules` is for managing project dependencies in the project. The `routes` folder provides the structured way to define how server handles incoming requests, resources, and middleware.

5.2.1 Models

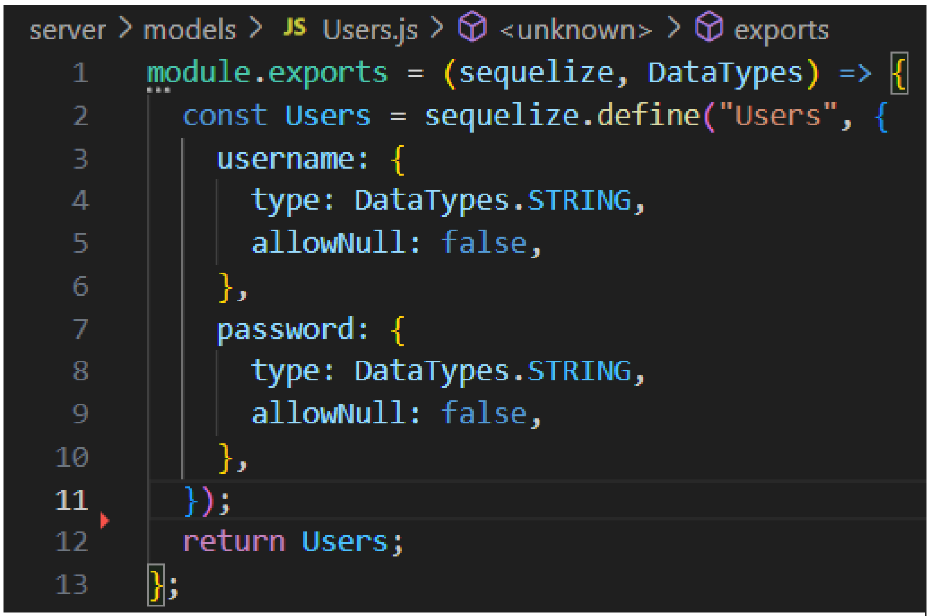


```
module.exports = (sequelize, DataTypes) => {
  // this basically generate the table "Machines" in mySQL
  const Machines = sequelize.define("Machines", {
    machine: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    function: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    username: {
      type: DataTypes.STRING,
      allowNull: false,
    },
  });
  Machines.associate = (models) => {
    Machines.hasMany(models.Descriptions, {
      onDelete: "cascade",
    });
  };

  return Machines;
};
```

Figure 2. Machine model

Figure 2 presents code snippet for a JavaScript module that builds a Sequelize model for dealing with the MySQL database table `Machines`. The `Machines` table is stored in MySQL DBMS and it is presented in Figure 11.

- First line `module.exports` exports a function with two parameters: sequelize and DataTypes. This function is the Sequelize model definition.
- sequelize.define function defines the model "Machines" in Sequelize. Function accepts two arguments: the model's name ("Machines") and an object that describes the model's attributes.
- machine - attribute represents the machine's name and is specified as a string. Attribute is configured to reject null values (allowNull: false), meaning the field must be modified when creating or updating a record.
- function - attribute represents the machine's function and is specified as a string. Like "machine", it does not support null values.
- username - attribute represents the machine's username. It is also a string type that rejects null values.
- Machines.associate function in Sequelize defines relationships between models. In this case, it indicates that a "Machines" object might have many "Descriptions" objects linked with it.
- Code Machines.hasMany(models.Descriptions, {onDelete: "cascade"});: creates a one-to-many relationship between the "Machines" and "Descriptions" models. This means that any machine can be assigned several descriptions.
- The {onDelete: "cascade”} option guarantees that when a machine is deleted, all related descriptions are likewise erased.
- return Machines: Finally, the method returns the "Machines" model, which can be imported and utilized throughout the program.

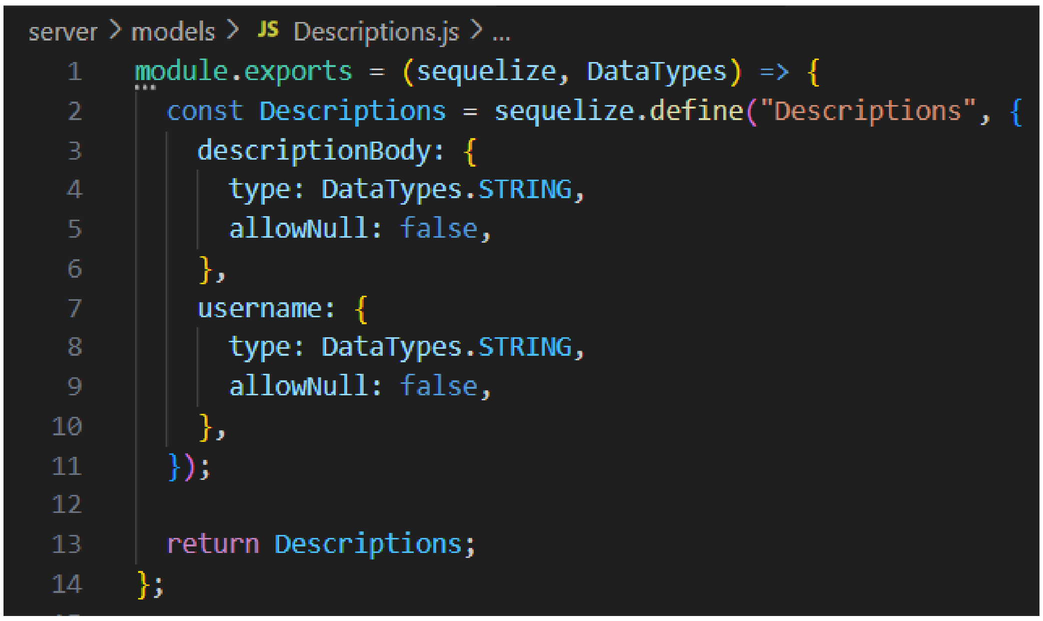


```
module.exports = (sequelize, DataTypes) => {
  const Users = sequelize.define("Users", {
    username: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    password: {
      type: DataTypes.STRING,
      allowNull: false,
    },
  });
  return Users;
};
```

Figure 3. Users model

Figure 3 presents code snippet for defining a Sequelize model for a database table named `Users` and this table is presented in Figure 12.

- module.exports exports the function with two parameters similarly as other models.
- sequelize.define function describes the model named `Users` in Sequelize.
- `username` and `password` - attributes are defined by an object with a `type` property specifying the data and an `allowNull` property specifying whether the attribute can be null or not.
- The function returns the `Users` model, making the application available for use.


server > models > JS Descriptions.js > ...

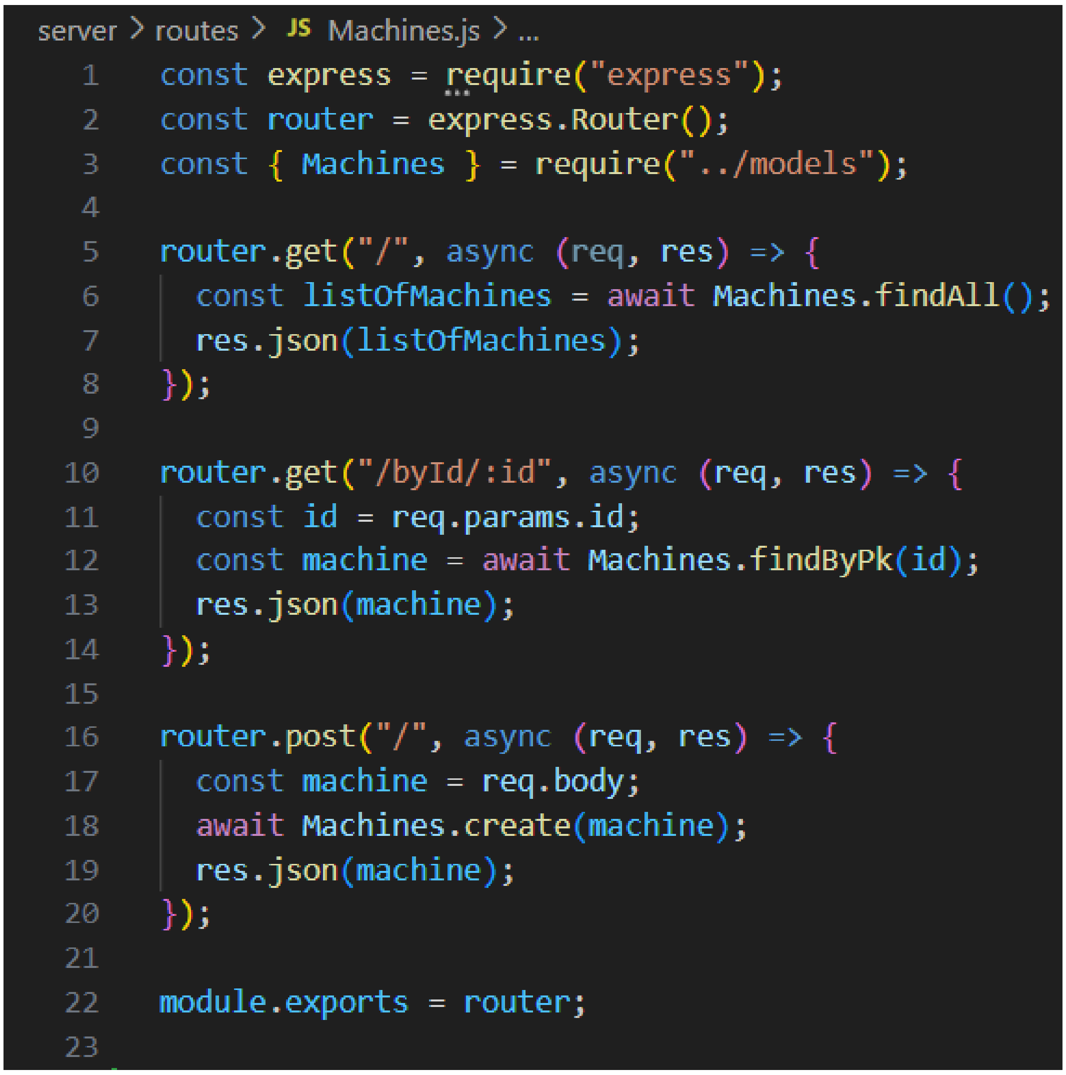
```
module.exports = (sequelize, DataTypes) => {
  const Descriptions = sequelize.define("Descriptions", {
    descriptionBody: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    username: {
      type: DataTypes.STRING,
      allowNull: false,
    },
  });

  return Descriptions;
};
```

Figure 4. Descriptions model

Figure 4 presents code snippet for defining a Sequelize model called “Descriptions” to deal with the database table named “Descriptions” which is presented in Figure 10.

- module.exports exports the function that takes two parameters.
- sequelize.define function takes two parameters, and an object defining the model’s attributes and data types.
- descriptionBody - attribute represents the body of the description with the data type String and not null.
- username - attribute also represents the username associated with the description, data type and requirement.
- Finally, Sequelize model definition is returned.

5.2.2 Routes



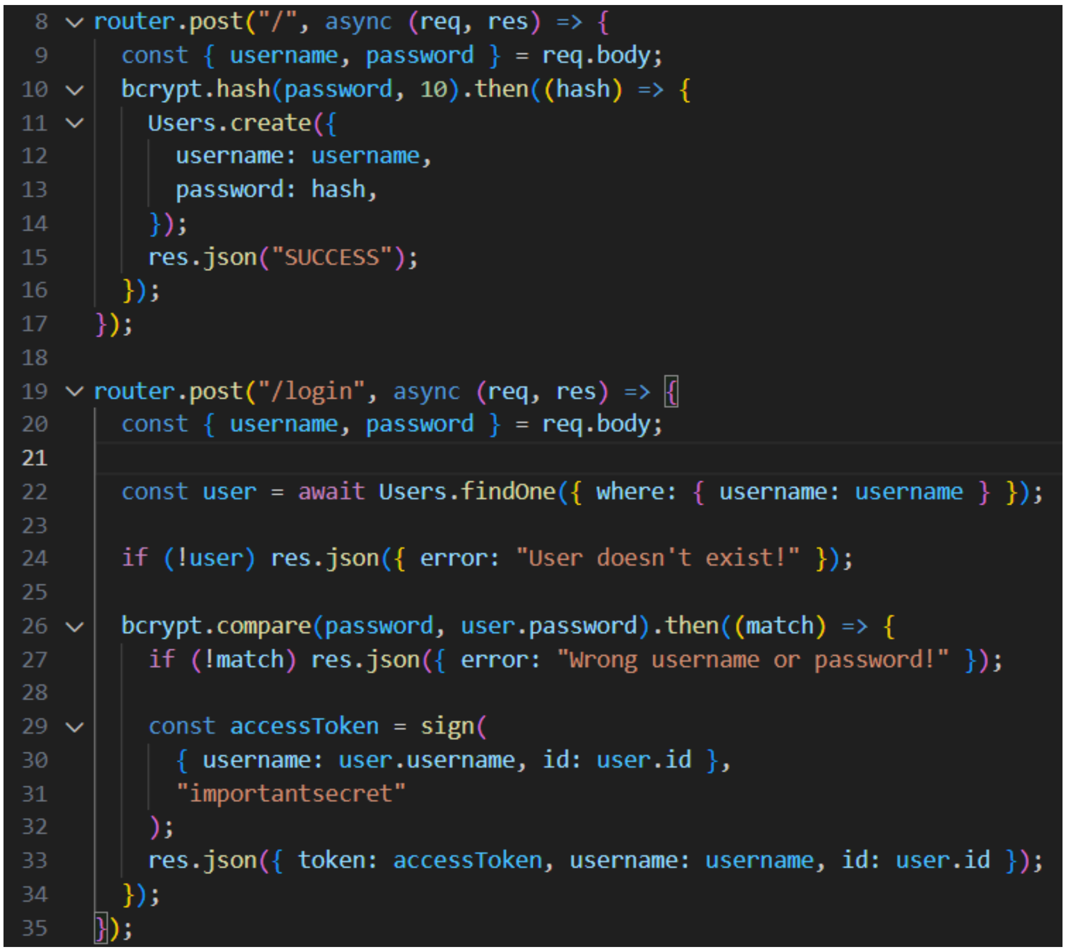
```javascript
const express = require("express");
const router = express.Router();
const { Machines } = require("../models");

router.get("/", async (req, res) => {
  const listOfMachines = await Machines.findAll();
  res.json(listOfMachines);
});

router.get("/byId/:id", async (req, res) => {
  const id = req.params.id;
  const machine = await Machines.findByPk(id);
  res.json(machine);
});

router.post("/", async (req, res) => {
  const machine = req.body;
  await Machines.create(machine);
  res.json(machine);
});

module.exports = router;
```

Figure 5. Machines route

The code shown in Figure 5 presents a part of a Node.js application that uses the Express.js framework. It defines a router that handles HTTP requests related to machines.

- express imports the Express.js framework, which makes it easier to create web applications and APIs using Node.js.

- router imports Express's Router module, which allows specifying routes individually and utilized in the application.
- {Machines} imports the Machines model from the ../models directory. Importing model establishes Sequelize model for machines and enables interaction with the database.
- GET `/` Route handles HTTP GET requests for the root URL ("/"). Calling the URL all machines from the database are retrieved using Sequelize's findAll() function and return data in json-format.
- GET `/byId/:id` Route replies to HTTP GET requests for URLs "/byId/:id", where:id is an URL argument that represents the machine's ID. Request fetches machine data from the database by its primary key (id) using Sequelize's findByPk() function and return data in json-format.
- POST `/` Route handles HTTP POST requests for the root URL ("/"). Request requires a JSON payload with machine data in the request body. Calling the URL then generates a new machine record in the database using Sequelize's create() function.
- All route handlers are declared as asynchronous functions using the `async` keyword. Async enables functions to utilize `await` to stop execution while asynchronous processes, like database queries or model building, are finished.
- For GET routes, the server returns JSON data describing the machines that were queried. The POST route, after generating a new machine record, the same machine data as JSON in the response is returned.



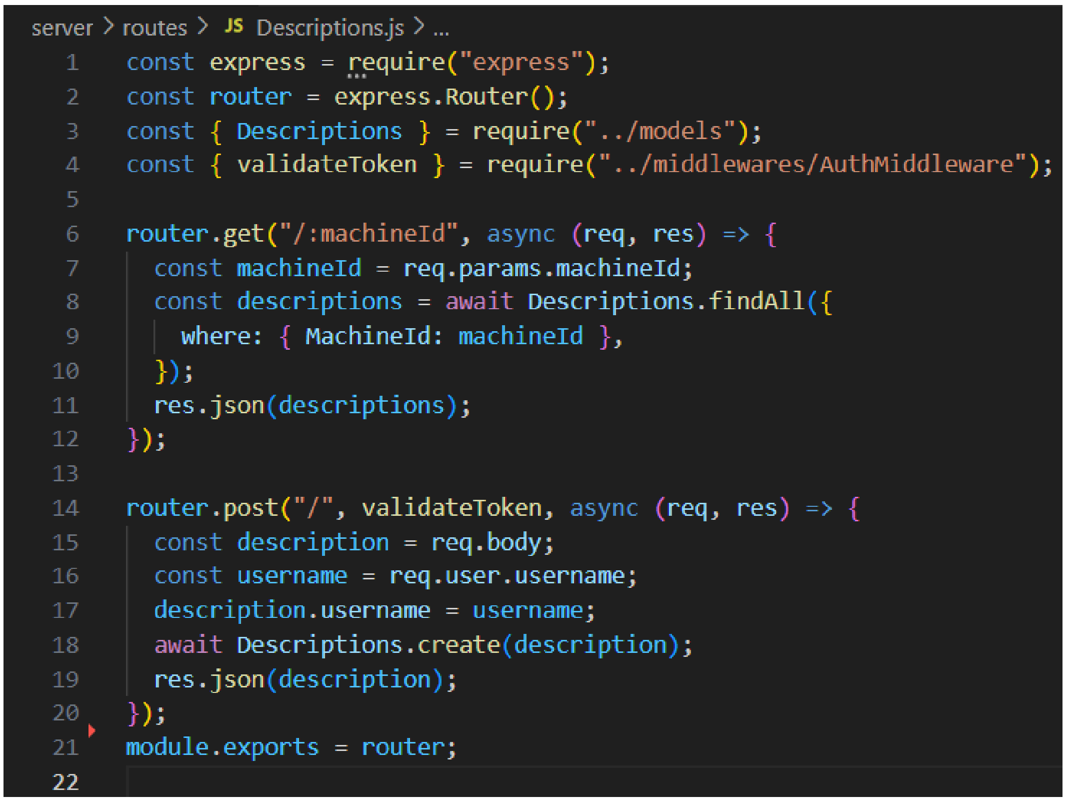
```
router.post("/", async (req, res) => {
  const { username, password } = req.body;
  bcrypt.hash(password, 10).then((hash) => {
    Users.create({
      username: username,
      password: hash,
    });
    res.json("SUCCESS");
  });
});

router.post("/login", async (req, res) => {
  const { username, password } = req.body;

  const user = await Users.findOne({ where: { username: username } });

  if (!user) res.json({ error: "User doesn't exist!" });

  bcrypt.compare(password, user.password).then((match) => {
    if (!match) res.json({ error: "Wrong username or password!" });

    const accessToken = sign(
      { username: user.username, id: user.id },
      "importantsecret"
    );
    res.json({ token: accessToken, username: username, id: user.id });
  });
});
```

Figure 6. Users route

Figure 6 presents the code for Users route, which implements basic user registration, login, and authentication functionality with Express, bcrypt for password hashing, and JWT for authentication. The code adheres to standard practices by securely hashing passwords and implementing JWTs for stateless authentication.

- Dependencies: express, bcrypt-hashing password securely, jsonwebtoken(JWT) and AuthMiddleware.
- Registration Route requires `username` and `password` in the request body, password is hashed using bcrypt.
- Authentication Route uses `validateToken`, a custom middleware, to authenticate requests. If the JWT token is valid, user information derived

from the token is returned. If the token is incorrect or absent, the middleware returns an error message.



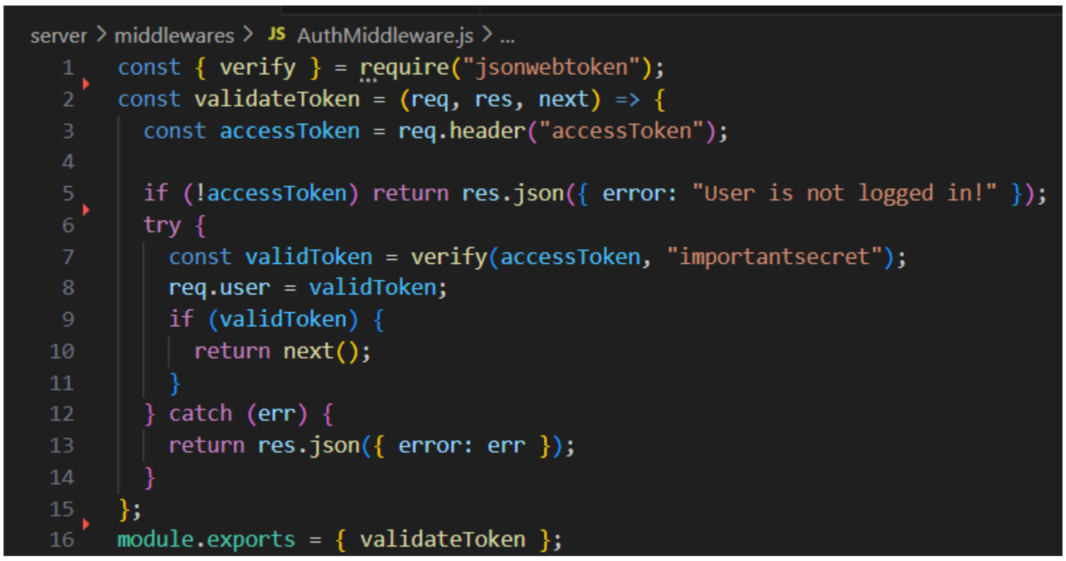
server > routes > JS Descriptions.js > ...

```javascript
const express = require("express");
const router = express.Router();
const { Descriptions } = require("../models");
const { validateToken } = require("../middlewares/AuthMiddleware");

router.get("/:machineId", async (req, res) => {
  const machineId = req.params.machineId;
  const descriptions = await Descriptions.findAll({
    where: { MachineId: machineId },
  });
  res.json(descriptions);
});

router.post("/", validateToken, async (req, res) => {
  const description = req.body;
  const username = req.user.username;
  description.username = username;
  await Descriptions.create(description);
  res.json(description);
});
module.exports = router;
```

Figure 7. Descriptions route

Figure 7 presents the code for Descriptions route using Express.js router module to performs CRUD (Create, Read, Update, and Delete) activities for a resource called "Descriptions." To conduct the activities, database communication is needed. Additionally, `express` is required to define a router with `express.Router()` for importing the Descriptions model and `validateToken` middleware from its respective modules.

5.2.3 Middleware



```javascript
const { verify } = require("jsonwebtoken");
const validateToken = (req, res, next) => {
  const accessToken = req.header("accessToken");

  if (!accessToken) return res.json({ error: "User is not logged in!" });
  try {
    const validToken = verify(accessToken, "importantsecret");
    req.user = validToken;
    if (validToken) {
      return next();
    }
  } catch (err) {
    return res.json({ error: err });
  }
};
module.exports = { validateToken };
```

Figure 8. Authentication middleware

The code snippet shown in Figure 8 provides the middleware function called `validateToken`. The function is for verifying the presence of a JWT (JSON Web Token) in the request header. If the token is present, authentication is done using a secret key. If the verification is successful, a decoded token is added to the request object and control is transferred to the next middleware or route handler. If the token is missing or incorrect, an error response is provided. This middleware function ensures that incoming requests are having a valid access token and information about the authenticated user is extracted for further processing in subsequent middleware functions or route handlers.

5.3 MySQL Database System Implementation

The project gives a demonstration of three types of data: Machine, User and Description. The Machine table contains properties machine, function, and username where the User table contains username and password, and the Description table contains descriptionBody and username.

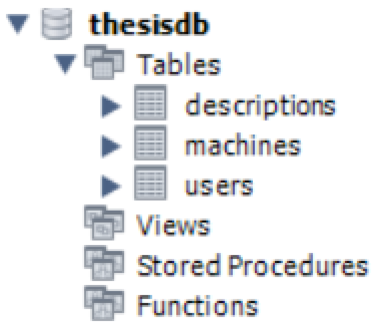
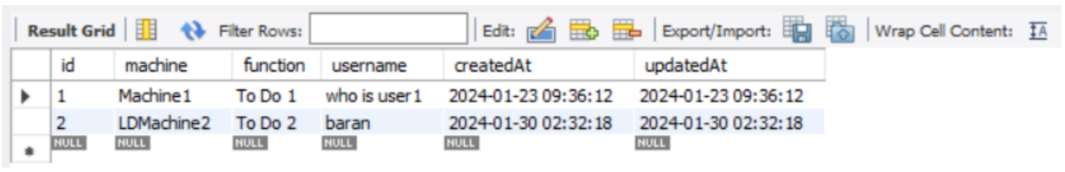


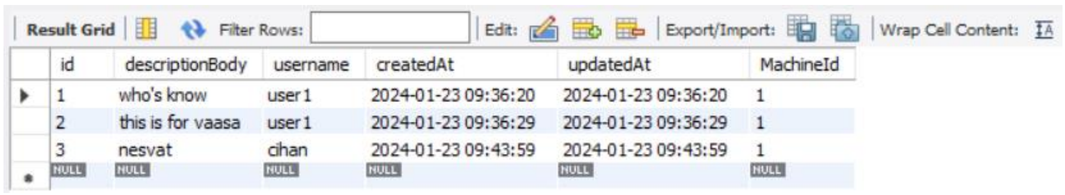
Figure 9. Database structure

Figure 9 represents the database schema for `thesisdb` -database which includes the mentioned three tables: descriptions, machines, and users. The schema also contains other objects called: Views, Stored Procedures and Functions. The `Views` object contains virtual tables derived from the result set of a select query. `Stored Procedures` are a set of SQL statements that are stored in the database and can be executed repeatedly by invoking their name. `Functions` are like stored procedures but return a single value. The ``Views` and `Stored Procedures` are not utilized in this project.



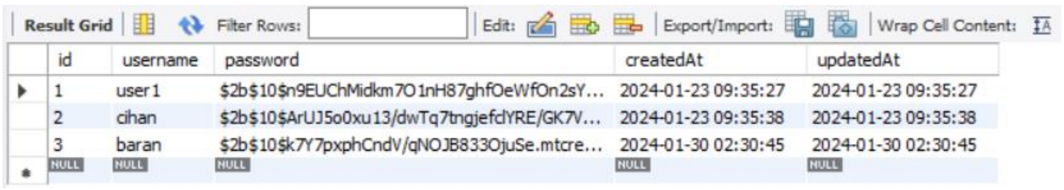
id	machine	function	username	createdAt	updatedAt
1	Machine1	To Do 1	who is user1	2024-01-23 09:36:12	2024-01-23 09:36:12
2	LDMachine2	To Do 2	baran	2024-01-30 02:32:18	2024-01-30 02:32:18
NULL	NULL	NULL	NULL	NULL	NULL

Figure 11. machines table in database



id	descriptionBody	username	createdAt	updatedAt	MachineId
1	who's know	user1	2024-01-23 09:36:20	2024-01-23 09:36:20	1
2	this is for vaasa	user1	2024-01-23 09:36:29	2024-01-23 09:36:29	1
3	nesvat	cihan	2024-01-23 09:43:59	2024-01-23 09:43:59	1
NULL	NULL	NULL	NULL	NULL	NULL

Figure 10. description table in database



id	username	password	createdAt	updatedAt
1	user1	$2b$10$n9EUChMidkm7O1nH87ghfOeWfOn2sY...	2024-01-23 09:35:27	2024-01-23 09:35:27
2	cihan	$2b$10$ArUJ5o0xu13/dwTq7tngjefclYRE/GK7V...	2024-01-23 09:35:38	2024-01-23 09:35:38
3	baran	$2b$10$k7Y7pxphCndV/qNOJB833OjuSe.mtcre...	2024-01-30 02:30:45	2024-01-30 02:30:45
NULL	NULL	NULL	NULL	NULL

Figure 12. users table in database

Figures 10, 11, 12 show the tables that store the data of the web application. In addition, to the fields that were described earlier, MySQL automatically generates id, createdAt and updatedAt for each table. The id field represents the unique identifier to an object (machine, user, or description) and two other fields store the time when the object has been modified.

5.4 Front-end Implementation

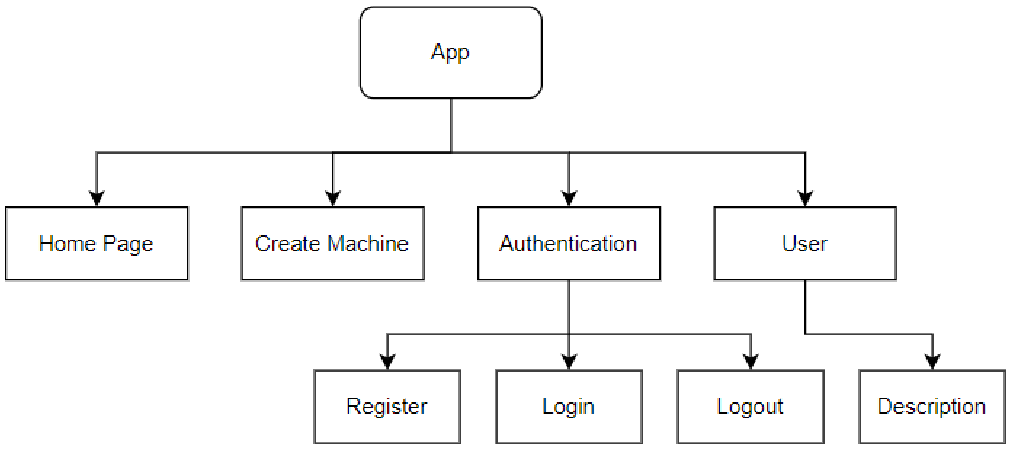


Figure 13. The Front-end structure

The graph in Figure 13 presents the structure of the web application developed in the thesis. The application has four main pages: Home page, Create machine page, Authentication page and User page. The Authentication page is run in three stages: a new user needs to create a new account with password in the Register step. After the account creation, the user can log in to the application. The User page has a description for giving information to the assigned machine.

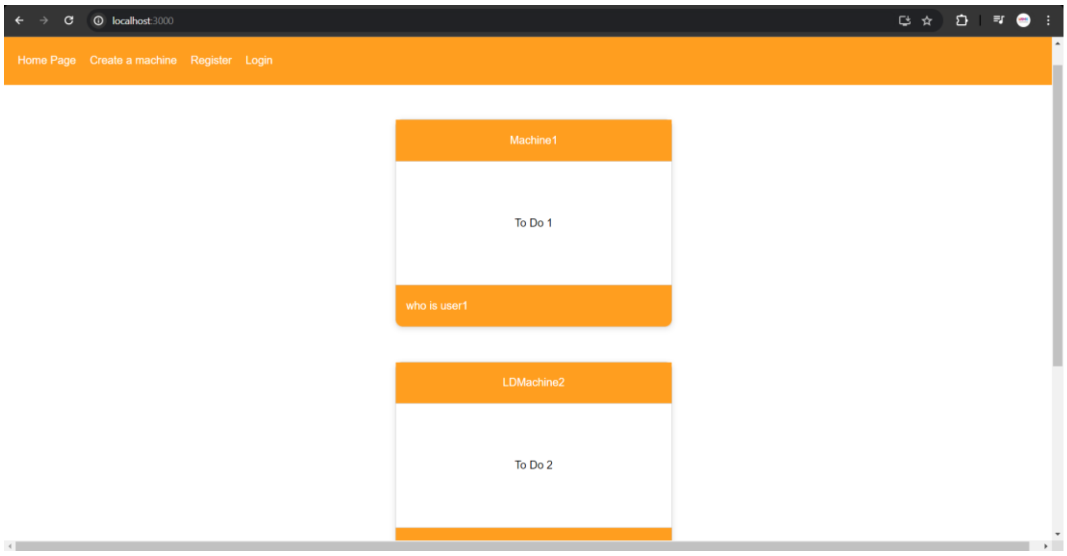


Figure 14. Home page

Figure 14 shows the Home page where all machines and their information are shown. Shown information includes machine name, functionality and username assigned for the machine. When the user clicks one of these blocks the machine is shown and the user can add more details about the machine.

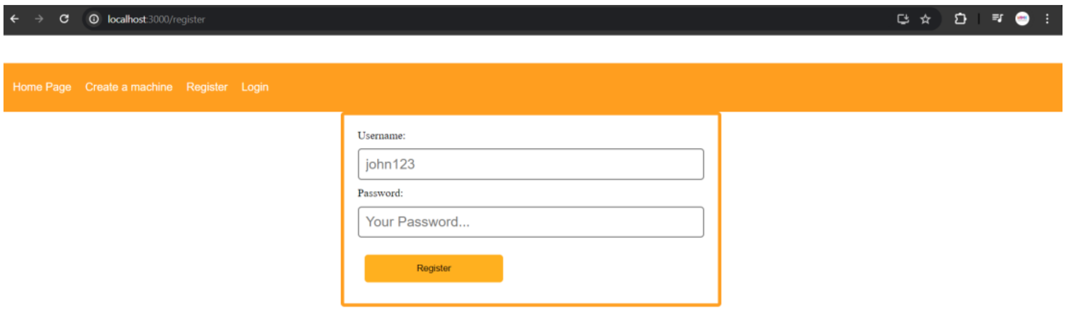


Figure 15. Register page

Figure 15 represents the registration process that has some requirements for the password. The password must be at least four characters long and it needs to contain special characters. After the registration step, the user can login to the application.

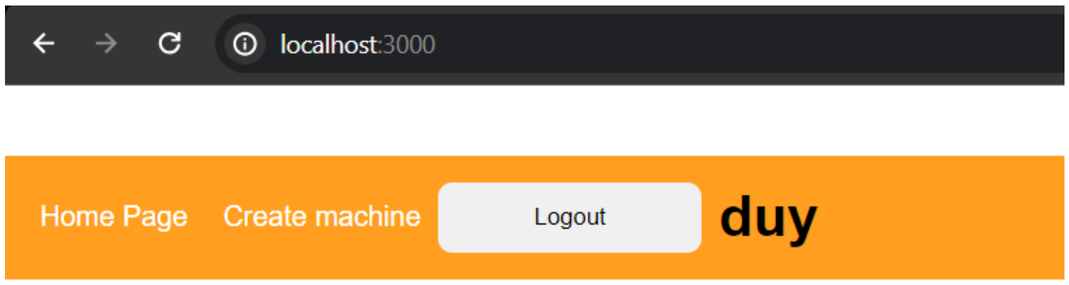


Figure 16. The button bar after login

Figure 16 presents the navigation bar after the user has logged in. The Register and Login buttons are hidden, the Logout button is displayed instead. Besides, the name of the user is shown to highlight user has logged in. The state of the navigation bar is reset to the initial process bar after Logout.

6 CONCLUSIONS

The creation of a full-stack web application is a substantial undertaking in the field of current software engineering. The study and analysis offered in this thesis show that such projects require a thorough grasp of both front-end and back-end technology, as well as the ability to smoothly integrate multiple components.

As stated throughout this paper, the advantage of full-stack web applications is necessary for web developers. They provide greater flexibility, scalability, and performance, resulting in a more coherent and dynamic digital experience. Furthermore, by embracing the most recent frameworks and technologies, developers may expedite the development process, lowering time-to-market and assuring ongoing relevance in an ever-changing technological context.

This application was developed using MySQL DBMS, Node.js, Express.js, React, Axios, and Sequelize. The project was tough but satisfying. During the first development phase, MySQL was used to design the database schema, while NodeJS and ExpressJS were set up for server-side programming. In the second phase the client side with React was built. During the development, many challenges were encountered, notably with the integration of several technologies, the user authentication method, and debugging and testing the application. The application's general architecture seems strong, and it offers a secure authentication method that protects users from illegal access.

To summarize, developing a full-stack web application is more than just a technical endeavor; it is also a demonstration of software engineers' intellect and imagination. Developers can realize the full potential of the web by adopting best practices, developing technologies, and prioritizing user-centric design, helping individuals and organizations to prosper in the digital era.

REFERENCES

1. Indeed. A Guide to Front-End vs. Back-End vs. Full-Stack Development. Accessed 1.02.2024. https://www.indeed.com/career-advice/finding-a-job/back-end-vs-front-end-vs-full-stack-development#:~:text=The%20back%20end%20refers%20to,of%20components%2C%20front%20and%20back
2. Simplilearn. What is Node.js: A Comprehensive Guide. Accessed 2.02.2024. https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs
3. MDN Web Docs. Express/Node introduction. Accessed 2.02.2024. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction
4. Medium. Node.js and Express.js: Powering Modern Web Development. Accessed 5.02.2024. https://medium.com/@IEEE_Computer_Society_VIT/node-js-and-express-js-powering-modern-web-development-b88eaa6cfc58
5. Turing. How to Create MySQL Connection with Node JS using Sequelize and Express. Accessed 5.02.2024. https://www.turing.com/kb/mysql-connection-with-node-js-using-sequelize-and-express
6. Kinsta. What Is Express.js? Everything You Should Know. Accessed 11.02.2024. https://kinsta.com/knowledgebase/what-is-express-js/
7. Hostinger. What Is MySQL and How Does It Work. Accessed 13.02.2024. https://www.hostinger.com/tutorials/what-is-mysql
8. Blueclawdb. MySQL Advantages and Disadvantages. Accessed 13.02.2024. https://blueclawdb.com/mysql/advantages-disadvantages-mysql/

9. IBM. Relational database structure. Accessed 13.04.2024. https://www.ibm.com/docs/en/mfci/7.6.2?topic=design-relational-data-base-structure
10. KDnuggets. Introduction to Databases in Data Science. Accessed 20.02.2024. https://www.kdnuggets.com/introduction-to-databases-in-data-science
11. PrepBytes. Trigger in DBMS. Accessed 20.02.2024. https://www.prep-bytes.com/blog/dbms/trigger-in-dbms/
12. Planetscale. What is MySQL and what is it used for? Accessed 23.02.2024. https://planetscale.com/learn/articles/what-is-mysql
13. Hubspot. What is React.js? Uses, Examples, & More. Accessed 29.02.2024. https://blog.hubspot.com/website/react-js
14. Tutorialspoint. ReactJS – Overview. Accessed 29.02.2024. https://www.tutorialspoint.com/reactjs/reactjs_overview.htm
15. Simplilearn. The best guide to know What is React. Accessed 29.02.2024. https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs
16. KnowledgeHut. What are the Pros and Cons of React. Accessed 3.03.2024. https://www.knowledgehut.com/blog/web-develop-ment/pros-and-cons-of-react#pros%C2%A0
17. Axios. Documentation What is Axios? Accessed 6.03.2024. https://axios-http.com/docs/intro
18. DigitalOcean. How to use Axios with React. Accessed 6.03.2024 https://www.digitalocean.com/community/tutorials/react-axios-react
19. Sanity. Form validation with Yup. Accessed 10.03.2024. https://www.san-ity.io/guides/form-validation-with-npm-yup