Ben-Gurion University
of the Negev

Exam Management System

Maintenance Guide

Students: Roi Tiefenbrunn, Ofek Nov, Idan Aharoni, Mor Abo

Customer: Dr. Mayer Goldberg

Contents

Introduction and Motivation

The Exam-Management-System is designed to create, manage, and improve exams.

By giving a web interface, inspection and maintenance of exams become faster easier, and more efficient.

The motivation behind this system is to reduce the manual workload, minimize errors, improve the quality of the questions, and improve the overall exam experience for both educators and students.

Maintenance

Language: JavaScript

Exam-Management-System is implemented using JavaScript only.

Server-Side Maintenance

Server: Restify

Restify is used as the server framework to handle API requests.

The server configuration, API endpoints, and middleware are all defined at: src/main/server.js

For more information about Restify

Authentication: JWT

JSON Web Tokens (JWT) are used for securing the API endpoints.

The JWT is transformed to the client on login and client request (see /refreshJWT endpoint).

If within an API request valid JWT token exists, The server will add to each request body an object: “callingUser” with the JWT information, if the JWT is not valid, the request will fail (except for /login, /logout /changePassword)

Database: PostgreSQL

The system is configured to work with Postgres.

Database Communication: Sequelize

Sequelize is an ORM for interacting with the PostgreSQL database.

For the schemas definitions head to the src/main/DAL, the repositories and data objects will be defined there.

For more information sequelize

Common Maintenance and Development Use Cases

Running the Server

- **Install dependencies:**

```
Cd src/main
Npm install
Npm run dev
```

Adding/Maintaining API Endpoints

- Add endpoint in src/main/server.js

```
server.post('/signUp', service.signUp);
```

- And handlers in src/main/business/applicationFacade.js

```
async register(userdetails){...}
```

Adding/Maintaining Models

- Define Sequelize models in the `src/main/DAL` directory.

```
function defineUserModel(sequelize) {...}
```

Authentication Implementation

- Adding information to the JWT: in the signIn handler

Compile text to latex:

- Different objects need to be compiled into latex, this could be done using the LatexCompiler, for example:

```
compileNormal(answer.text, this.#createCropCallback('5 5 5 5', callback));
```

Testing Backend

- **Run tests:**

```
Cd src
npm run test
```

Client-Side Maintenance

Frontend: React JS

The frontend of the Exam Management System is developed using React.js.

It leverages the Next.js framework. For more information, please refer to docs.

Common Maintenance and Development Use Cases

Running the Client

- **Install dependencies:**

```
cd ReactApp
npm install
```

- **Start the client:**

```
cd ReactApp
npm install
npm run dev
```

Adding/Maintaining Pages

- Define new pages and components in the /ReactApp/src/page, Due to .next, the page will be automatically added.
- Add a new component to the sidebar, via /ReactApp/src/layout/dashbord/config.js as follows:

```
{
    title: TITLE,
    path: '/page-path,
    icon: (
      <SvgIcon fontSize="small">
        <Newspaper />
      </SvgIcon>
    ),
    permissions: [types.WHO_HAS_PERMISSION]
  }
```

Network Requests

- Use The following functions from /ReactApp/src/utils/rest-api-call/js

```
export async function requestLatexServer(path, body) {...}
export async function requestServer(path, method, body) {...}
```

Those functions are connected to the server and will do the REST API calls.

PDF Generation using LaTeX

The system compiles exams into PDF format using a LaTeX server.

LaTeX server implementation

The latex server is based on functional programming, using callbacks to build the final exam pdf.

The main function is

```
compileNormal(latexCode, callback) {...}
```

This enables the addition of a new "latex object" to the system and dynamically builds the exam.

Possible Future Features

- Add more task types.
- Support Hebrew in the latex format.
- Grading student exams.
- Statistical analysis of student answers.
- Statistical analysis of keys/distractors.