

COSC3060 – 2024B – Exercises

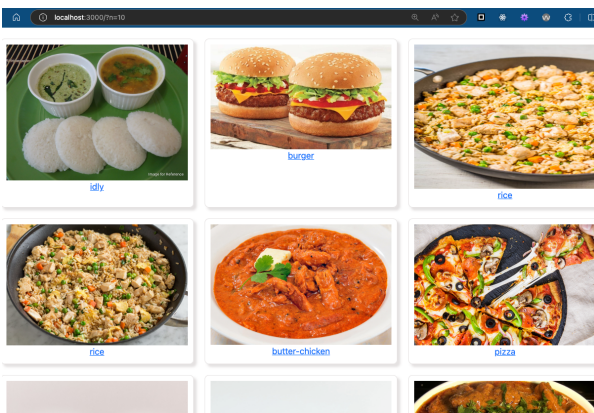
Before you start the exercise, please clone the github project

https://github.com/tuanrmit/w7_exercise

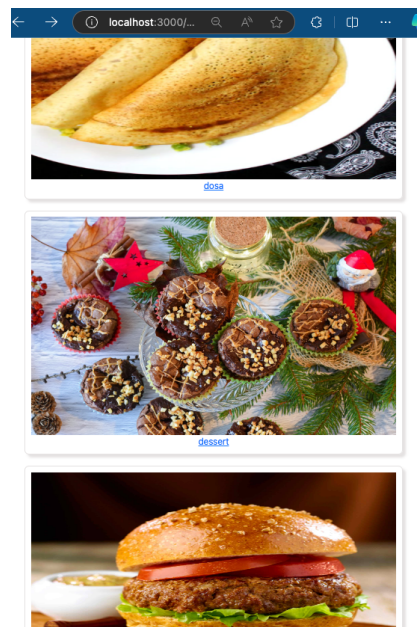
which contains the necessary files for the exercise!

Exercise 1.

Let's make a simple web application for retrieving and displaying a random number of food images by accessing the API <https://foodish-api.com>. The number of images to retrieve is specified in the query part of the URL.

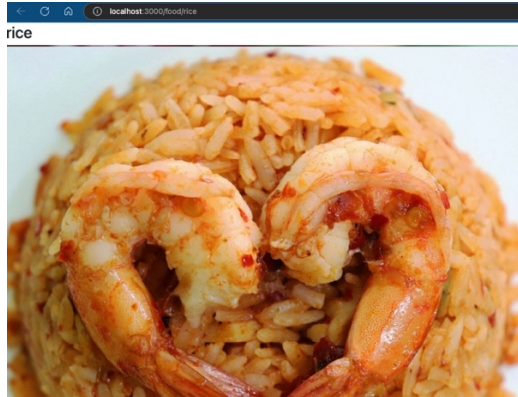


Wireframes for desktop displays (screen width $\geq 992\text{px}$)



Wireframes for small displays (screen width $< 992\text{px}$)

And clicking the link under the food image will open another tab with random image of the same type of food.



Packages/modules

Install all required packages. Do not use any dependencies other than those predefined in the "package.json".

Database

In the foodModel.js, implement the foodSchema. All data fields are required. Refer to the file seed.js and view its data to know the foodSchema structure

The mongoose.js file helps to connect the app with MongoDB Atlas. Modify the DB environment variable found in the .env file with your MongoDB Atlas connection string details. Make the app connect your MongoDB Atlas, to database of your choice.

Execute the seed.js to insert data into your database.

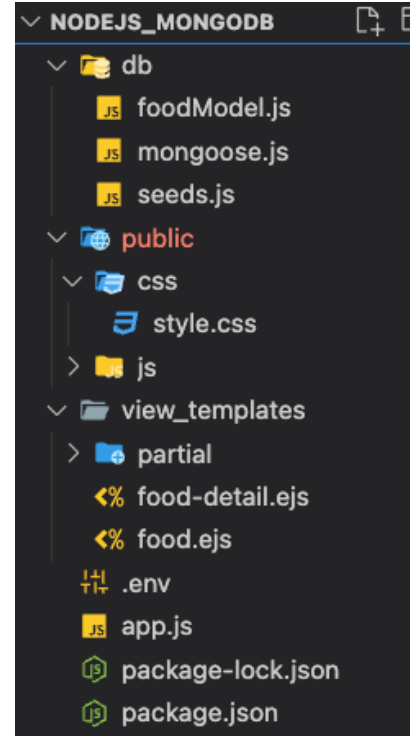
Server

Modify the .env and configure the app.js so that the port specified in the .env file is used for running the app. If that port is not available, using port 8888 instead.

Modify the code in "app.js" to launch the application. You may need to make some changes to the folder structure to resolve some bugs.

View engine/EJS

Configure the view engine so that .ejs files in the view_templates can be used to display content.



Configuration for convenient use

In the **package.json**, add **two scripts: "start" and "seed"**. The "start" script will launch the application, and the "seed" script will insert data into the database, eliminating the need to specify configurations with each execution. These commands will be performed like this:

```
○ → [red box] npm run start
```

```
● → [red box] npm run seed
```

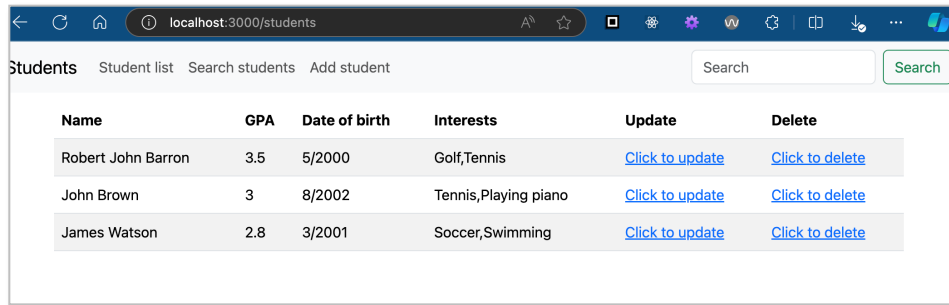
```
> node app.js
```

```
> node db/seed.js
```

Exercise 2.

Let make a simple application for managing students' information. This application helps to:

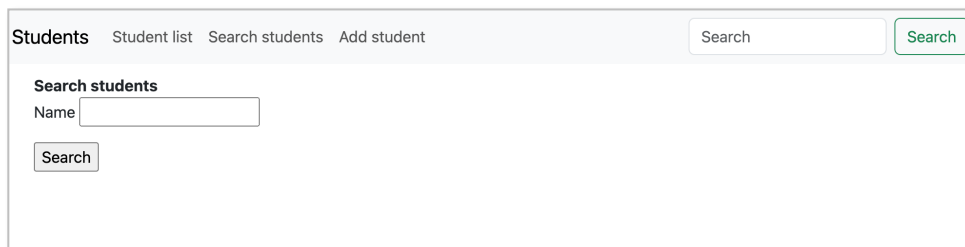
- list all students existing in a database,



The screenshot shows a web browser at localhost:3000/students. The application has a header with 'Students', 'Student list', 'Search students', and 'Add student'. A search bar is on the right. Below is a table with columns: Name, GPA, Date of birth, Interests, Update, and Delete. The table contains three rows of student data.

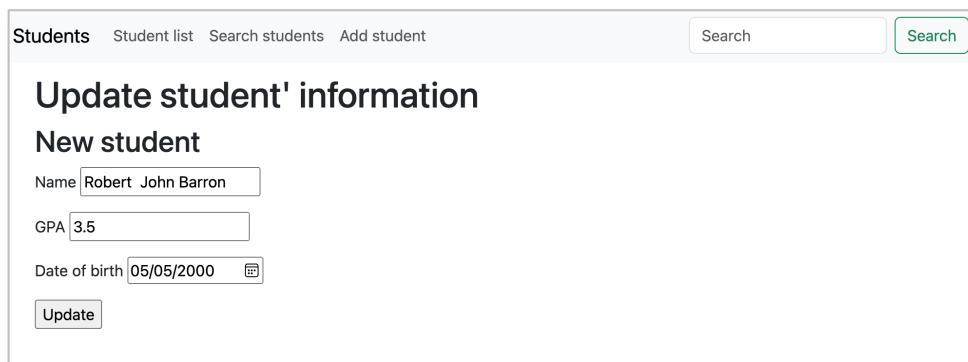
Name	GPA	Date of birth	Interests	Update	Delete
Robert John Barron	3.5	5/2000	Golf,Tennis	Click to update	Click to delete
John Brown	3	8/2002	Tennis,Playing piano	Click to update	Click to delete
James Watson	2.8	3/2001	Soccer,Swimming	Click to update	Click to delete

- search for student information by student name,



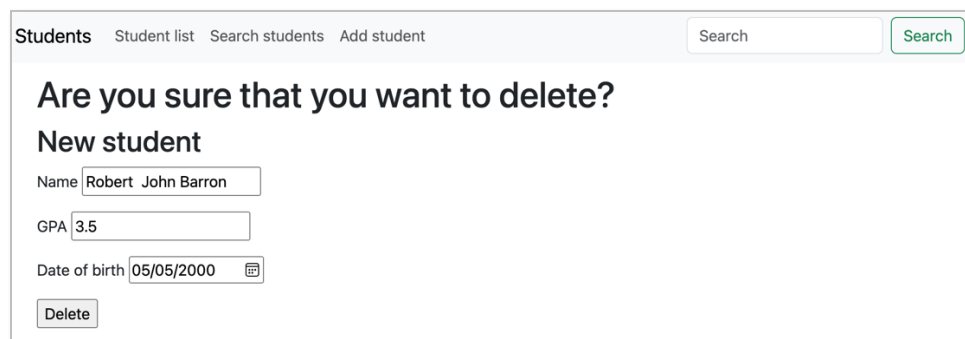
The screenshot shows the 'Search students' form. It has a header with 'Students', 'Student list', 'Search students', and 'Add student'. A search bar is on the right. Below is a form with a label 'Search students', a text input for 'Name', and a 'Search' button.

- update students' information,



The screenshot shows the 'Update student' form. It has a header with 'Students', 'Student list', 'Search students', and 'Add student'. A search bar is on the right. Below is a form with the title 'Update student' and 'New student'. It has text inputs for 'Name' (Robert John Barron), 'GPA' (3.5), and a date input for 'Date of birth' (05/05/2000). There is an 'Update' button.

- and delete student records.



The screenshot shows the 'Delete student' form. It has a header with 'Students', 'Student list', 'Search students', and 'Add student'. A search bar is on the right. Below is a form with the title 'Are you sure that you want to delete?' and 'New student'. It has text inputs for 'Name' (Robert John Barron), 'GPA' (3.5), and a date input for 'Date of birth' (05/05/2000). There is a 'Delete' button.

Packages/modules

Install all required packages. Do not use any dependencies other than those predefined in the "package.json".

Database

In the student.js, implement the studentSchema. All data fields are required. Refer to the file seed.js and view its data to know the foodSchema structure

The mongoose.js file helps to connect the app with MongoDB Atlas. Modify the DB environment variable found in the .env file with your MongoDB Atlas connection string details. Make the app connect your MongoDB Atlas, to database of your choice.

Execute the seed.js to insert data into your database.

Server

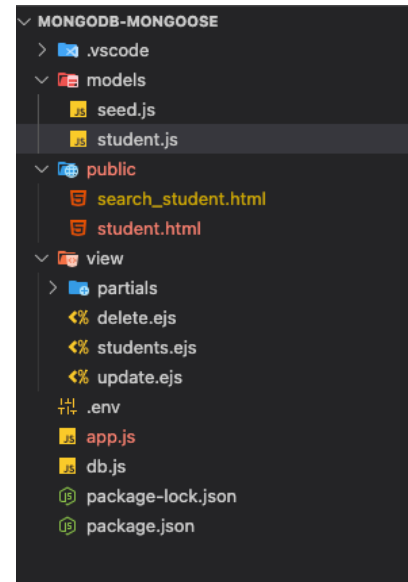
Modify the .env and configure the app.js so that the port specified in the .env file is used for running the app. If that port is not available, using port 8888 instead.

Modify the code in "app.js" to launch the application. You may need to make some changes to the folder structure to resolve some bugs.

Configure app.js so that static files in public folder is accessible.

View engine/EJS

Configure the view engine so that .ejs files in the view can be used to display content.



https://github.com/tuanrmit/w7_exercise