MEAN (MongoDB, Express.js, Angular, NodeJS) Stack Project Setup Guide

Submitters: Liad Uziel & Ohad Cohen

This guide will walk you through the steps to set up and run our MEAN stack project. The project consists of a frontend built with Angular, a backend using Node.js, and a MongoDB database.

Prerequisites

Before you begin, make sure you have the following prerequisites installed on your system:

- **Node.js**: The runtime environment for running JavaScript applications.

- **Angular CLI**: A command-line tool for Angular development.

- **MongoDB**: A NoSQL database for storing data.

Installation Steps

1. Install Node.js

Node.js is required for both the frontend and backend development. Follow these steps to install Node.js:

1. Visit the https://nodejs.org.

2. Download the LTS (Long-Term Support) version suitable for your operating system.

3. Run the installer and follow the installation prompts.

To verify that Node.js and npm (Node Package Manager) are installed, open your terminal and run the following commands:

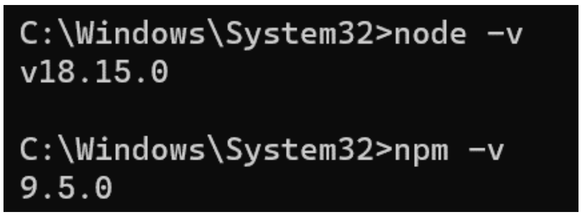
node -v

npm -v

```
node -v
npm -v
```

You should see the installed Node.js and npm versions.

These are the Node and NPM versions that ran locally on our computer when we developed the project:



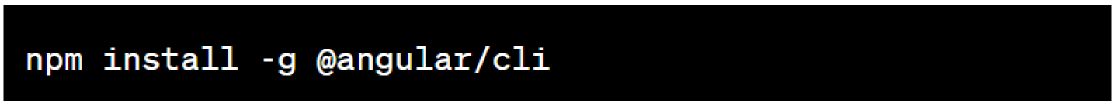
```
C:\Windows\System32>node -v
v18.15.0

C:\Windows\System32>npm -v
9.5.0
```

2. Install Angular CLI

Angular CLI is used for managing Angular projects. To install it, open your terminal and run the following command:

npm install -g @angular/cli



```
npm install -g @angular/cli
```

Verify the installation by checking the Angular CLI version:

ng --version

```
ng --version
```

3. Install MongoDB

MongoDB is our database system. You can download and install it from the https://www.mongodb.com/try/download/community

Follow the installation instructions for your specific operating system.

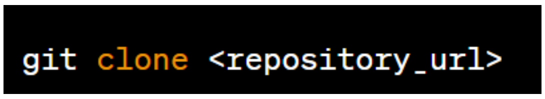
Project Setup

Now that you have installed the prerequisites, it's time to set up our project.

1. Clone the Project Repository

Clone the project's repository from the provided link:

git clone <repository_url>



```
git clone <repository_url>
```

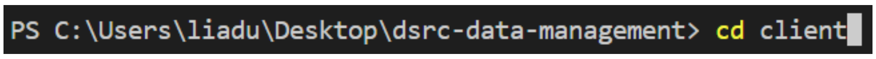
Replace `<repository_url>` with the actual URL of your project's repository.

Repository_url: https://gitlab.com/u_of_haifa_is_dpt/capstone_mor_peleg/dsrc-data-management.git

2. Frontend Setup

Navigate to the project's frontend directory when you are inside the folder you cloned.:

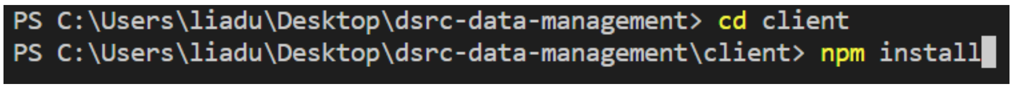
cd client



```
PS C:\Users\liadu\Desktop\dsrc-data-management> cd client
```

Install the project's frontend dependencies:

npm install

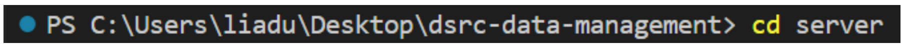


```
PS C:\Users\liadu\Desktop\dsrc-data-management> cd client
PS C:\Users\liadu\Desktop\dsrc-data-management\client> npm install
```

3. Backend Setup

Navigate to the project's backend directory:

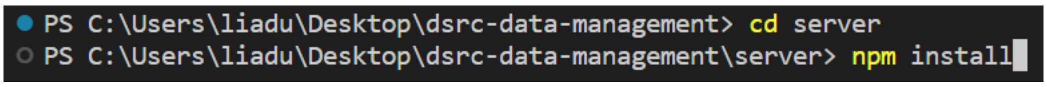
cd server



```
PS C:\Users\liadu\Desktop\dsrc-data-management> cd server
```

Install the project's backend dependencies:

npm install

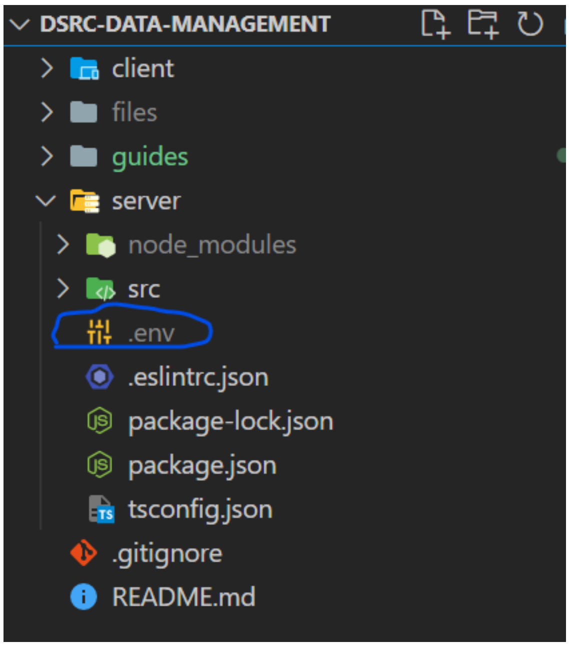


```
PS C:\Users\liadu\Desktop\dsrc-data-management> cd server
PS C:\Users\liadu\Desktop\dsrc-data-management\server> npm install
```

In addition, we have an .env file for the mongo address, jwt secret key, and application email details that sends the emails.

This is a file that is not in the repository because it is in git ignore because it contains information that should not be revealed to everyone, mostly for security reasons.

So it is necessary to put it in the root folder of the backend project.



```
DSRC-DATA-MANAGEMENT
  > client
  > files
  > guides
  v server
    > node_modules
    > src
      .env
      .eslintrc.json
      package-lock.json
      package.json
      tsconfig.json
    .gitignore
    README.md
```

4. Configure MongoDB

Make sure MongoDB is running on your system.

We managed our data in the database with the help of Mongo DB Compass

If you don't have this software, install it from here:

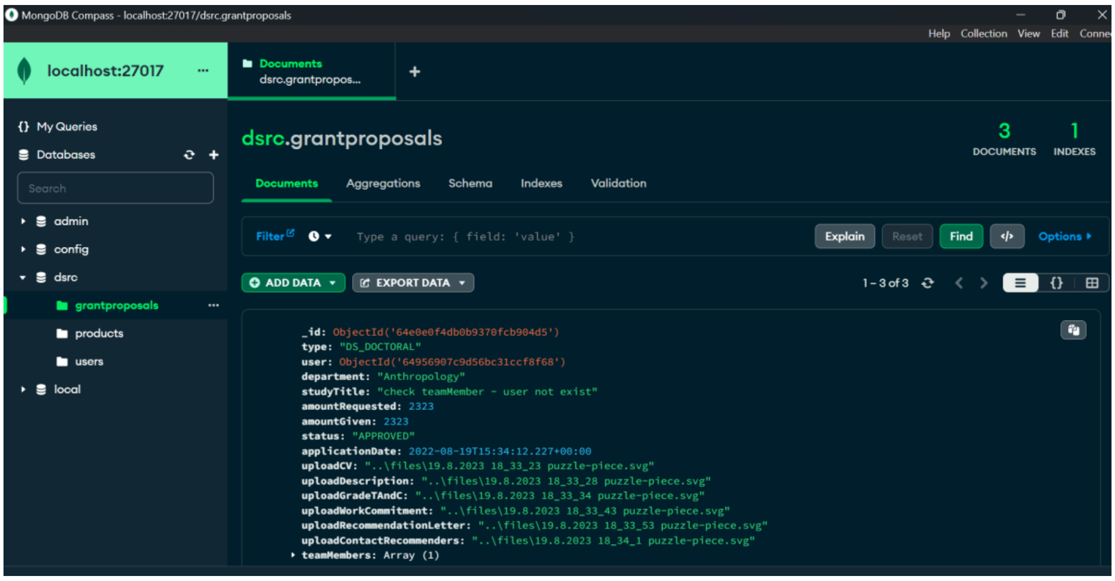
https://www.mongodb.com/try/download/compass

We used the gui version:
TOOLS

MongoDB Compass Download (GUI)

.

This is how mongo db compass of our project looks:

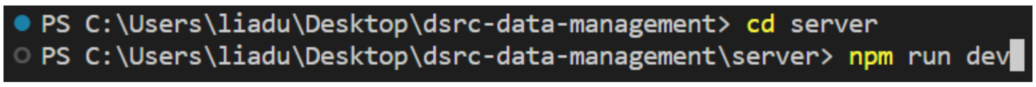

MongoDB Compass - localhost:27017/dsrc.grantproposals
Help Collection View Edit Conne
localhost:27017
Documents dsrc.grantpropos...
My Queries
Databases
Search
admin
config
dsrc
grantproposals
products
users
local
dsrc.grantproposals
3 DOCUMENTS
1 INDEXES
Documents Aggregations Schema Indexes Validation
Filter
Type a query: { field: 'value' }
Explain Reset Find
Options
ADD DATA
EXPORT DATA
1 – 3 of 3

```
_id: ObjectId('64e0e0f4db0b9370fcb904d5')
type: "DS_DOCTORAL"
user: ObjectId('64956907c9d56bc31ccf8f68')
department: "Anthropology"
studyTitle: "check teamMember - user not exist"
amountRequested: 2323
amountGiven: 2323
status: "APPROVED"
applicationDate: 2022-08-19T15:34:12.227+00:00
uploadCV: "..\files\19.8.2023 18_33_23 puzzle-piece.svg"
uploadDescription: "..\files\19.8.2023 18_33_28 puzzle-piece.svg"
uploadGradeTAndC: "..\files\19.8.2023 18_33_34 puzzle-piece.svg"
uploadWorkCommitment: "..\files\19.8.2023 18_33_43 puzzle-piece.svg"
uploadRecommendationLetter: "..\files\19.8.2023 18_33_53 puzzle-piece.svg"
uploadContactRecommenders: "..\files\19.8.2023 18_34_1 puzzle-piece.svg"
teamMembers: Array (1)
```

5. Run the Backend

To start the backend server, run the following command from the project's backend directory:

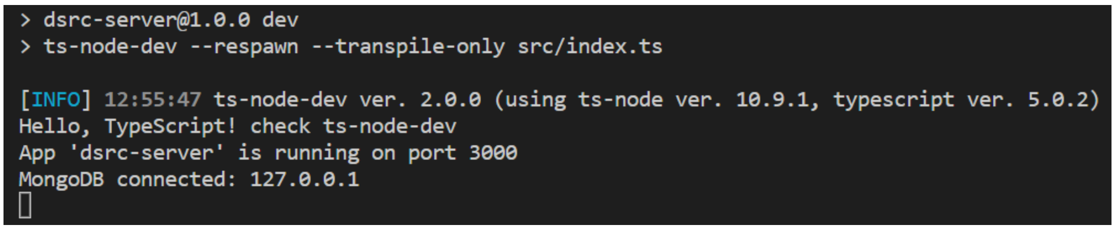
npm run dev



```
PS C:\Users\liadu\Desktop\dsrc-data-management> cd server
PS C:\Users\liadu\Desktop\dsrc-data-management\server> npm run dev
```

The backend should now be running at `http://localhost:3000/`.

This is how you know that the backend project is running successfully



```
> dsrc-server@1.0.0 dev
> ts-node-dev --respawn --transpile-only src/index.ts

[INFO] 12:55:47 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.1, typescript ver. 5.0.2)
Hello, TypeScript! check ts-node-dev
App 'dsrc-server' is running on port 3000
MongoDB connected: 127.0.0.1
```

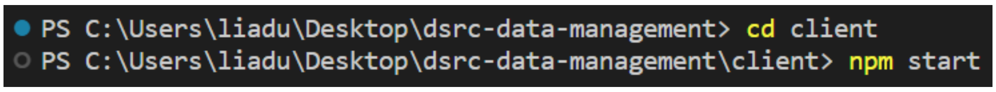
Accessing the Application

You can access the application by opening your web browser and navigating to `http://localhost:4200/`. The frontend will communicate with the backend, which serves the necessary data from the MongoDB database.

6. Run the Frontend

To start the frontend server, run the following command from the project's frontend directory:

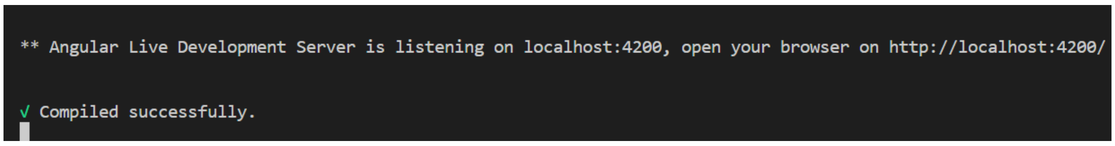
npm start / ng serve



```
PS C:\Users\liadu\Desktop\dsrc-data-management> cd client
PS C:\Users\liadu\Desktop\dsrc-data-management\client> npm start
```

The frontend should now be accessible at ` http://localhost:4200/ ` in your web browser.

This is how you know that the frontend project is running successfully



```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/

√ Compiled successfully.
```

Explanation of each library

Frontend Libraries

1. lodash

- Purpose: lodash is a utility library that provides a wide range of helpful functions for working with arrays, objects, and more.
- Use in Frontend: lodash can simplify tasks such as data manipulation, filtering, and iteration within your Angular frontend code.

2. lottie

- Purpose: Lottie is a library that allows you to easily render and play After Effects animations in your web application.
- Use in Frontend: You may have used Lottie to add dynamic and visually appealing animations to your user interface, enhancing the user experience.

3. rxjs (Reactive Extensions for JavaScript)

- Purpose: RxJS is a library for handling asynchronous operations and events using Observables and operators.
- Use in Frontend: In Angular, RxJS is commonly used for handling HTTP requests, managing data streams, and implementing reactive patterns in your frontend code.

4. ngx-toastr

- Purpose: ngx-toastr is a library for displaying toast notifications in Angular applications.
- Use in Frontend: It's used to show informative and user-friendly notifications (e.g., success messages, error alerts) to users as they interact with your application.

5. primeng

- Purpose: PrimeNG is a UI component library for Angular applications, providing a rich set of pre-built UI components and themes.

- Use in Frontend: You likely used PrimeNG components to build various parts of your frontend user interface, saving development time and ensuring consistency in design.
  Most of the component in the project is primeng includes: Table, dropdown, menubar, text, text area, password, email and etc.

6. Angular Material

- Purpose: Angular Material is a UI component library for Angular applications, providing a rich set of pre-built UI components and themes.

- Use in Backend: used for password strength in register page.

Backend Libraries

1. bcrypt

- Purpose: bcrypt is a library for hashing and salting passwords, enhancing security by protecting user credentials.

- Use in Backend: You probably used bcrypt to securely hash and verify passwords for user authentication in your Node.js backend.

2. dotenv

- Purpose: dotenv is a library for loading environment variables from a \`.env\` file into your Node.js application.

- Use in Backend: It helps manage sensitive configuration details, such as database connections and API keys, by storing them in environment variables.

3. express

- Purpose: Express.js is a fast, minimalist web framework for Node.js, simplifying the creation of robust and scalable web applications.

- Use in Backend: Express is the foundation of your Node.js backend, handling HTTP requests, routing, and middleware to create the API endpoints of your application.

4. jsonwebtoken (JWT)

- Purpose: jsonwebtoken is a library for generating and verifying JSON Web Tokens (JWTs), used for authentication and secure data transmission.

- Use in Backend: JWTs are likely used to authenticate and authorize users, ensuring secure access to your backend API.

5. mailgen

- Purpose: mailgen is a library for generating HTML email templates for sending transactional emails.

- Use in Backend: You may have used mailgen to create visually appealing and standardized email notifications within your application.

6. nodemailer

- Purpose: nodemailer is a library for sending email from Node.js applications.
- Use in Backend: Nodemailer allows you to implement email functionality, such as sending registration confirmation emails or password reset links.

7. mongoose

- Purpose: mongoose is an Object-Data Modeling (ODM) library for MongoDB, simplifying database operations and schema management.
- Use in Backend: Mongoose facilitates interactions with MongoDB, including data modeling, validation, and querying.

8. multer

- Purpose: multer is a middleware for handling multipart/form-data, commonly used for file uploads.
- Use in Backend: If your application allows users to upload files, multer is likely used to handle and process file uploads in your Node.js backend.

9. TypeScript

- Purpose: TypeScript is a superset of JavaScript that adds static typing and other features, enhancing code quality and maintainability.
- Use in Backend: TypeScript provides type safety and improved code documentation in your Node.js backend, making development more efficient and less error-prone.

These libraries play crucial roles in your MEAN stack project, enhancing functionality, security, and user experience in both the frontend and backend components of your application.