GRIFFITH COLLEGE DUBLIN

Assignment Cover Sheet

Student name:	**Ali Benjouad**		
Student number:	**3952766**		
Faculty:	**Computing Science**		
Course:	**Computer science**	**Stage/year:**	**3rd year**
Subject:	**Web Technologies**		
Study Mode:	Full time **X**	Part-time	
Lecturer Name:	**Gemma Deery**		
Assignment Title:	**Assigmnent 04**		
No. of pages:			
Disk included?	Yes	No	
Additional Information:	(ie. number of pieces submitted, size of assignment, A2, A3 etc)		
Date due:	**19/05/2024**		
Date submitted:	**19/05/2024**		

Plagiarism disclaimer:

I understand that plagiarism is a serious offence and have read and understood the college policy on plagiarism. I also understand that I may receive a mark of zero if I have not identified and properly attributed sources which have been used, referred to, or have in any way influenced the preparation of this assignment, or if I have knowingly allowed others to plagiarise my work in this way.

I hereby certify that this assignment is my own work, based on my personal study and/or research, and that I have acknowledged all material and sources used in its preparation. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of anyone else, including other students.

Signed: _____ **Date:** _____

Please note: *Students **MUST** retain a hard / soft copy of **ALL** assignments as well as a receipt issued and signed by a member of Faculty as proof of submission.*

Directory Structure:

ROOT FOLDER: angular_04

```
/angular_04/frontend
 ├── src
|   ├── app
|   |   ├── app-routing.module.ts
|   |   ├── app.component.css
|   |   ├── app.component.html
|   |   ├── app.component.spec.ts
|   |   ├── app.component.ts
|   |   ├── app.config.server.ts
|   |   ├── app.config.ts
|   |   ├── app.routes.ts
|   |   ├── app.server.module.ts
|   |   ├── components
|   |   |   ├── answers
|   |   |   |   ├── answers.component.css
|   |   |   |   ├── answers.component.html
|   |   |   |   ├── answers.component.spec.ts
|   |   |   |   └── answers.component.ts
|   |   |   ├── auth
|   |   |   |   ├── auth.component.css
|   |   |   |   ├── auth.component.html
|   |   |   |   ├── auth.component.spec.ts
|   |   |   |   └── auth.component.ts
|   |   |   |   ├── login.component.css
|   |   |   |   ├── login.component.html
|   |   |   |   ├── login.component.spec.ts
|   |   |   |   ├── login.component.ts
|   |   |   |   ├── register.component.css
|   |   |   |   ├── register.component.html
|   |   |   |   ├── register.component.spec.ts
|   |   |   |   ├── register.component.ts
|   |   |   ├── comments
|   |   |   |   ├── comments.component.css
```

```
|   |   |   |   ├── comments.component.html
|   |   |   |   ├── comments.component.spec.ts
|   |   |   |   └── comments.component.ts
|   |   |   ├── questions
|   |   |   |   ├── questions.component.css
|   |   |   |   ├── questions.component.html
|   |   |   |   ├── questions.component.spec.ts
|   |   |   |   └── questions.component.ts
|   |   |   ├── not-found
|   |   |   |   ├── not-found.component.css
|   |   |   |   ├── not-found.component.html
|   |   |   |   ├── not-found.component.spec.ts
|   |   |   |   └── not-found.component.ts
|   |   ├── services
|   |   |   ├── answer.service.spec.ts
|   |   |   ├── answer.service.ts
|   |   |   ├── auth.service.spec.ts
|   |   |   ├── auth.service.ts
|   |   |   ├── comment.service.spec.ts
|   |   |   ├── comment.service.ts
|   |   |   ├── question.service.spec.ts
|   |   |   └── question.service.ts
|   ├── environments
|   |   ├── environment.ts
|   |   └── environment.prod.ts
|   ├── assets
|   ├── favicon.ico
|   ├── index.html
|   ├── main.server.ts
|   ├── main.ts
|   ├── polyfills.ts
|   ├── styles
|   |   └── bootstrap.min.css.map
|   └── styles.css
├── angular.json
├── package-lock.json
├── package.json
```

```
├── proxy.conf.json
├── README.md
└── server
    ├── config
    │   ├── database.js
    │   └── passport.js
    ├── controllers
    │   ├── answerController.js
    │   ├── commentController.js
    │   ├── questionController.js
    │   └── userController.js
    ├── logs
    │   ├── combined.log
    │   ├── error.log
    │   └── exceptions.log
    ├── models
    │   ├── Answer.js
    │   ├── Comment.js
    │   ├── Profile.js
    │   ├── Question.js
    │   └── User.js
    ├── package-lock.json
    ├── package.json
    ├── routes
    │   ├── answers.js
    │   ├── comments.js
    │   ├── questions.js
    │   └── users.js
    ├── server.js
    └── utils
        ├── logger.js
        └── validation.js
```

Breakdown of the scripts (add as you go):

Frontend:

1- **app-routing.module.ts:**

- Handles routing configurations for the application.
- Includes routes for various components like login, register, questions, answers, comments, and a wildcard route for 404 Not Found.

2- **app.component.*** (all of the app.component scripts in frontend/src/app)

- The root component for the Angular application.
- Contains basic HTML structure and a router outlet.

•

3- **components/**

- Contains various components organized by feature: answers, auth, comments, questions, not-found

4- **services/**

- Contains service classes for handling HTTP requests to the backend API, the backend API is proven to be working using the following command from /frontend for example:”

- curl -X POST http://localhost:5001/api/users/register -H "Content-Type: application/json" -d '{"name": "Test User", "email": "test@example.com", "password": "password"}':
- Traceback for the above:

“

server % node server.js

2024-05-19 22:47:24 info: Connecting to MongoDB...
2024-05-19 22:47:24 info: Server running on port 5001
(node:83737) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
2024-05-19 22:47:24 info: MongoDB connected successfully
2024-05-19 22:49:13 info: Incoming request: POST /api/users/register
2024-05-19 22:49:13 info: RegisterUser endpoint called with data:
2024-05-19 22:49:13 warn: Registration failed: Email already exists

“

- curl -X POST http://localhost:5001/api/users/login -H "Content-Type: application/json" -d '{"email": "test@example.com", "password": "password"}'
- Traceback of the above:

“

2024-05-19 22:50:26 info: Incoming request: POST /api/users/login
2024-05-19 22:50:26 info: LoginUser endpoint called for:
2024-05-19 22:50:26 info: User logged in successfully:

“

The API Backend routes are working fine, as I derived the implementation from a previous working version, into integrating it witht the front-end

5- environments/

- Contains environment configuration files for different deployment environments

6- index.html

- The main HTML file for the application.
- Contains the <base> tag and the root <app-root> component.

7- main.ts

- The main entry point for the Angular application.
- Bootstraps the root module.

8- polyfills.ts

- Includes polyfills needed for Angular applications to support different browsers.

9- styles.css

- Global styles for the application.

Backend:

1- server/config/

- Contains configuration files for database and authentication.

2- server/controllers/

- Contains controller files that handle business logic for different API endpoints.

3- server/logs/

- Contains log files logged by utils

4- server/models/

- Contains Mongoose models for database collections.

5- server/routes/

- Contains route files that define API endpoints and link them to controller functions.

6- server/server.js

- The main server file.
- Configures and starts the Express server.
- Sets up middleware, database connection, and routes.

7- server/utils/

- Contains utility functions and modules.

Breakdown of Frontend Components:

LoginComponent (login.component.ts):

- Handles user login functionality.
- Contains methods for submitting login form and navigating to register page.
- Includes ngOnInit lifecycle hook for initialization tasks.

RegisterComponent (register.component.ts)

- Handles user registration functionality.
- Contains methods for submitting registration form and navigating to login page.

NotFoundComponent (not-found.component.ts)

- Displays a 404 error message for unknown routes.

Services

AuthService (auth.service.ts):

- Provides methods for user authentication (login and register).
- Uses HttpClient to make HTTP requests to the backend API.

Troubleshooting phase

The entire project's progress was halted because of an error that just did not make a lot of sense, I have gathered as much information regarding the error from the research I have done to the all of the methods that I have tried to fix it, but still no change done resulted in a fix

This analysis is based off research, then I applied it

1. **Error Message**: NG0303: Can't bind to 'ngIf' since it isn't a known property of 'p'.
- **Cause**: The *ngIf directive is not recognized because the CommonModule is not imported in the module where the component is declared.
- **Fix**: Ensure CommonModule is imported in AppModule.

2. **Routing Issues**:
- **Cause**: Incorrect or missing routes in app-routing.module.ts.
- **Fix**: Ensure all necessary routes are defined, including a wildcard route for 404 Not Found.

3. **HTTP Requests Not Reaching Backend**:
- **Cause**: Incorrect URL paths or CORS issues.
- **Fix**: Verify API endpoint URLs and ensure CORS is configured correctly on the server.

4. **Console Errors**:
- **Cause**: Various issues such as missing files, incorrect bindings, or configuration problems.
- **Fix**: Address each specific error by reviewing stack traces and verifying configurations.

Methodologies Used to Troubleshoot:

Ensuring Correct Module Imports:

- Verified that CommonModule is imported in the AppModule.
- Double-checked all other module imports to ensure they were correct.

Routing Verification:

- Ensured all routes, including wildcard for 404, are correctly defined in app-routing.module.ts.
- Added additional logging to the routing module to trace navigation events.

API Endpoint Verification:

- Used curl to test API endpoints and confirmed they work independently.
- Verified CORS configuration on the server to ensure requests from the frontend are accepted.

Console Logging:

- Added console logs to track the execution flow in components and services.
- Included additional logs to capture HTTP request and response details.

Error Analysis:

- Analyzed and addressed errors in the browser console and backend logs.
- Implemented a global error handler in Angular to catch and log all uncaught errors.

Fallback Mechanism:

- Added verbose logging in both frontend and backend to trace any missed issues.
- Enabled Angular's production mode to catch any configuration issues not visible in development mode.

But none of these changes seemed to fix the error:”
[Error] Failed to load resource: the server responded with a status of 404 (Not Found) (bootstrap.min.css.map, line 0)
[Error] NG0303: Can't bind to 'ngIf' since it isn't a known property of 'p' (used in the 'LoginComponent' component template).

If the 'ngIf' is an Angular control flow directive, please make sure that either the 'NgIf' directive or the 'CommonModule' is a part of an @NgModule where this component is declared.

reportUnknownPropertyError (vendor.js:28020)
handleUnknownPropertyError (vendor.js:28014)
elementPropertyInternal (vendor.js:34012)
ɵɵproperty (vendor.js:37664)
LoginComponent_Template (LoginComponent.js:65)
executeTemplate (vendor.js:33596)
refreshView (vendor.js:35020)
detectChangesInView (vendor.js:35182)
detectChangesInComponent (vendor.js:35160)
detectChangesInChildComponents (vendor.js:35194)
refreshView (vendor.js:35070)
detectChangesInView (vendor.js:35182)
detectChangesInEmbeddedViews (vendor.js:35129)
refreshView (vendor.js:35043)
detectChangesInView (vendor.js:35182)
detectChangesInComponent (vendor.js:35160)
detectChangesInChildComponents (vendor.js:35194)
refreshView (vendor.js:35070)
detectChangesInternal (vendor.js:34963)
detectChanges (vendor.js:35470)
tick (vendor.js:50025)
(anonymous function) (vendor.js:50188)
onInvoke (vendor.js:32685)
run (polyfills.js:8259)
next (vendor.js:50187)
next (vendor.js:5815)
_next (vendor.js:5784)
next (vendor.js:5757)
(anonymous function) (vendor.js:5594)
errorContext (vendor.js:8137)
next (vendor.js:5587)
emit (vendor.js:32269)
checkStable (vendor.js:32606)
onHasTask (vendor.js:32701)
hasTask (polyfills.js:8519)
_updateTaskCount (polyfills.js:8539)
_updateTaskCount (polyfills.js:8396)

runTask (polyfills.js:8319)
drainMicroTaskQueue (polyfills.js:8672)

“

Which is a crucial step as the bing is the link to the API of the backend and the rest of the routes as well, therefore I have tested the backend and it works independently but the front end remained blocked because of this error.

Research References:

- Angular documentation on CommonModule (2021). Retrieved from Angular CommonModule Documentation:(“https://angular.io/api/common/CommonModule “)

- Express documentation on middleware (2021). Retrieved from Express Middleware Documentation:(“ https://expressjs.com/en/guide/using-middleware.html “)

- **Angular ngIf directive**: The ngIf directive in Angular conditionally includes a template based on the value of an expression. It's a structural directive that adds or removes elements from the DOM. For more details, refer to the Angular documentation on ngIf Directive. :(“ https://www.w3schools.com/angular/ng_ng-if.asp “)

- **Angular CommonModule**: The CommonModule is a fundamental Angular module that exports basic directives and pipes such as ngIf and ngFor. It's automatically included in every feature module created. For comprehensive information, see the CommonModule documentation. :(“ https://angular.io/api/common/CommonModule “)

- **Angular routing configuration**: Angular's router enables navigation between views or components. It uses a configuration array to define routes and can handle lazy loading, guards, and resolvers. Detailed guidance can be found in the Angular Router Guide. :(“ https://angular.io/guide/router “)

- **Angular router-outlet**: The router-outlet directive acts as a placeholder where the router dynamically loads the components based on the current router state. More information is available in the Angular Router Guide. :(“ https://angular.io/guide/router#router-outlet “)

- **Angular lifecycle hooks**: Lifecycle hooks in Angular (like ngOnInit, ngOnChanges) allow you to tap into key moments in the lifecycle of a component or directive. For a complete list and usage, refer to the Angular Lifecycle Hooks. :(“ https://angularindepth.com/posts/1494/complete-guide-angular-lifecycle-hooks “)

- **Angular error handling**: Angular provides various ways to handle errors, such as using HttpInterceptor for HTTP errors or defining global error

handlers. For detailed practices, check the Angular Error Handling Guide. :(“ https://yon.fun/angular-error-handle/ “)

Angular development vs production mode: Angular offers different configurations for development and production. The production mode includes optimizations like AOT compilation and minification, while the development mode provides more debug information. See the Angular Deployment Guide for more information. :(“ https://angular-book.dev/ch04-07-development-and-production-builds.html

- “)
https://angular-book.dev/ch04-07-development-and-production-builds.html

- **Angular console logging best practices**: Effective logging in Angular can be achieved using services that wrap around console.log, allowing for centralized control of logging levels and output formats. Refer to Angular Best Practices for guidelines. :(“ “)

- **Angular 404 not found component configuration**: Configuring a 404 component in Angular involves defining a wildcard route that directs to a custom "Not Found" component. More information can be found in the Angular Router Guide. :(“ https://angular.io/guide/router “)