Group 6 MedAccess – Karthik Rammoorthy

Individual Reflection and Evaluation

Group Formation and Operation

Our group consisted of four members. Since we know each other already, it was easy for us to collaborate. We decided to implement an online doctor appointment portal after several brainstorming sessions. We named the project MedAccess. Each team member involved in one component of the Requirement Gathering phase. I researched the suitable technology stack to be considered for the project. Initially, we had some difficulty in assigning roles, as various group members had skills that could have been applied in many parts of the project. As the project specification became more clear, however, assigning roles also became easier, as it was clear that our skill-sets made us suited to particular parts of the project. Once the project started progressing, we functioned very well as a group. Also, we developed a test plan that was needed to be incorporated to assist in Test Driven Development. We also developed the Project Design Document that acted as a guide in the Development phase.

Decision on Technology and Project Architecture

I was eager to develop a full stack web project. A front-end framework was communicating with the backend framework through REST API requests. I discussed it with the team, and I was fortunate that the team was ready to implement it. We were enthralled to choose Angular 6 because of its powerful UI and JavaScript functionalities. With Typescript, object-oriented UI programming provides powerful capabilities. With regards to Backend, there was a discussion whether to implement.Net or Java. Since all the other team members were comfortable working with Java, we decided to develop a Backend with Java. Once I am starting learning about Java Springboot Framework. I tried hosting the backend in Heroku, and it worked like a charm. So, we decided to implement our project in Angular frontend with Springboot REST API hosted in Heroku Server.

Work Breakdown

We followed the agile methodology for this project. Every team member had the opportunity to be Scrum Master for one week. At the start of every sprint, we planned the list of activities to be performed for that week. At the end of the week, we evaluated our progress. Trello was useful for managing the Agile activities. I was the Scrum master on several occasions. It was a learning curve where I got to know how to manage the team in an agile environment.

Contribution to Project

We were new to Angular and Test Driven Development. So we struggled a lot during the initial stages. My roles and responsibilities are as follows,

Setting up work Environment

- I faced real difficulties in setting up the work environment.
- It took several days of work for me just to set up the Angular CLI project with Material Design for UI and FlexLayout API for responsive design.
- Once the setup is done, I loaded the project in Git, and team members pulled from it to work on their part. Similarly, it was my responsibility to set up backend as well.
- I developed a basic Springboot API with controller, model, service and DAO components and shared it with the team.

Hosting in Heroku

- Once the project setup was completed, we planned to load backend and frontend separately in Heroku. It was the most challenging work of the project.
- I was able to host backend in Heroku and consumed the REST endpoints in the frontend. Similarly, the frontend is also hosted in Heroku as well.

Search Bar Component

- One of my responsibilities is to develop the Search Bar component of the project. I created a search bar using Angular.
- In the backend, I created the individual controller with Doctor model and DAO objects.
- The search bar compares the keyword with doctor first and last names and displays in the search result page.
- I wrote the unit test cases for the search component during development, and it helped me to write the code efficiently.
- I followed SOLID principles while coding. I created an interface for every class to maintain cohesion among the systems.

Register Component

- Designed Angular forms with asynchronous validation.
- Built corresponding business logic and deployed as a REST endpoint from backend
- Send the validated data to the backend via REST API with JSON.
- Performed unit tests on the component.

Configurable Business Logic

- I created validation rules for registration component such as the minimum number of digits for a password, zip code pattern, etc.
- All these validation rules are stored in the database as a table.
- The register component in frontend extracts the validation rules employs it for validation.

- The valid input is sent back to the backend for the insertion process.

Asynchronous Functioning

- Angular provides us features to develop a single page and progressive web application.
- All the components are modular with its CSS, HTML and Typescript classes. For the components to communicate and react dynamically, Subject-Observer pattern is implemented.
- This helped a lot for login functionality where menu bar component observes when login component performs login operation.

Error Debugging during Code Development

- I started the project earlier than my team. Therefore, I faced many issues while development; when my team faced similar issues, I was able to pitch in and sort the issue.
- As I started to solve any error, I became increasingly confident about the technology.

Contribution By Classes

S No	Component	Method	Description
Front End Components – MedAccess Project			
1	Project Setup	-	Did brainstorming and setup basic working Angular CLI project to act as front end framework and shared with team.
2	App-routing.module.ts	Typescript file	This act as router for different components for the single page application.
3	Register Component	• HTML file • CSS file • Typescript file	Designed UI elements with validation. Applied CSS and wrote code to send POST request to REST endpoint with JSON data.
4	Searchbar Component	• HTML file • CSS file • Typescript file	Added search bar with navigation in order to do keyword search.
5	Searchresult Component	• HTML file • CSS file • Typescript file	Displays list of doctors matching the keyword.
6	Datasharing.service.ts	Typescript file	An injectable file that stores behavioural objects that keeps track of user

			login and updates variables.
7	Interceptor.module.ts	Module file	Intercepts HTTP requests to solve CORS issue and logs HTTP requests to backend API
8	Log.service.ts	Typescript file	This is responsible for front end logging where logs are recorded in different categories such as debug, error, info and fatal.
9	Rest.service.ts	createPatient, getDoctorById, getDoctorByName, getBusinessLogic	Communicates with Backend REST API.
10	Server.js	JS file	An Express JS file that hosts the Angular as a standalone application to be hosted in Heroku.
Back End Components - MedAccessAPI			
S No	**Class**	**Methods**	**Description**
1	WebConfig	• Com.MedAccess.Utils Folder. • addCorsMapping()	In order to solve CORS issue
2	Setup Project	-	Developed a basic Java SpringbootAPI package and shared in Github with team.
3	Doctor	• Com.MedAccess.Model Folder.	Doctor Model class with variables and getter-setter properties.
4	Patient	• Com.MedAccess.Model Folder.	Patient Model class with variables and getter-setter properties.
5	BusinessLogic	• Com.MedAccess.Model Folder.	BusinessLogic Model class with variables and getter-setter properties.
6	IPatientDao	• Com.MedAccess.Dao Folder.	Interface for PatientDao
7	PatientDao	• Com.MedAccess.Dao Folder. • getPatentById() • createPatient() • updatePatient() • deletePatient() • getAllPatients()	

8	DoctorDao	Com.MedAccess.Dao Folder. getDoctorByName()	
9	PatientDaoTest	Unit Test methods for DoctorDao class.	
10	IBusinessLogicDao	Com.MedAccess.Dao Folder.	Interface for BusinessLogicDao
11	BusinessLogicDao	Com.MedAccess.Dao Folder. getBusinessLogic()	Retrieves businesslogic from database using stored procedure
12	IPatientService	Com.MedAccess.Service Folder.	Interface for Patient Service
13	PatientService	Com.MedAccess.Service Folder. • getPatentById() • createPatient() • updatePatient() • deletePatient() • getAllPatients()	
14	DoctorService	Com.MedAccess.Service Folder. getDoctorByName()	
15	IBusinessLogicService	Com.MedAccess.Service Folder.	Interface for BusinessLogicService
16	BusinessLogicService	Com.MedAccess.Service Folder. getBusinessLogic()	
17	Config files	Application.properties Dbconfig.xml	Configuration files
18	PatientController	• getPatentById() • createPatient() • updatePatient() • deletePatient() • getAllPatients()	RESTful endpoints to be consumed by front end
19	DoctorController	getDoctorByName()	
20	BusinessLogicController	getBusinessLogic()	

Reflective Evaluation

This was one the best groups I have worked on. Each team member was aware of their roles and responsibilities. When one of the members stuck in any issue, others helped and solved it. On a personal side, I learned a lot through the course. Rob, being a veteran in Industry taught us how to write a code the Industry expects us to write. One thing our team could have changed is the technology that we chose. We would have chosen a pure Java application without REST API and front-end framework so that we would have implemented more design patterns. Full stack project ate more work hours. On the whole, it was a great experience, and I am confident that the knowledge I gained would help me a lot for cracking job placements and internship interviews.