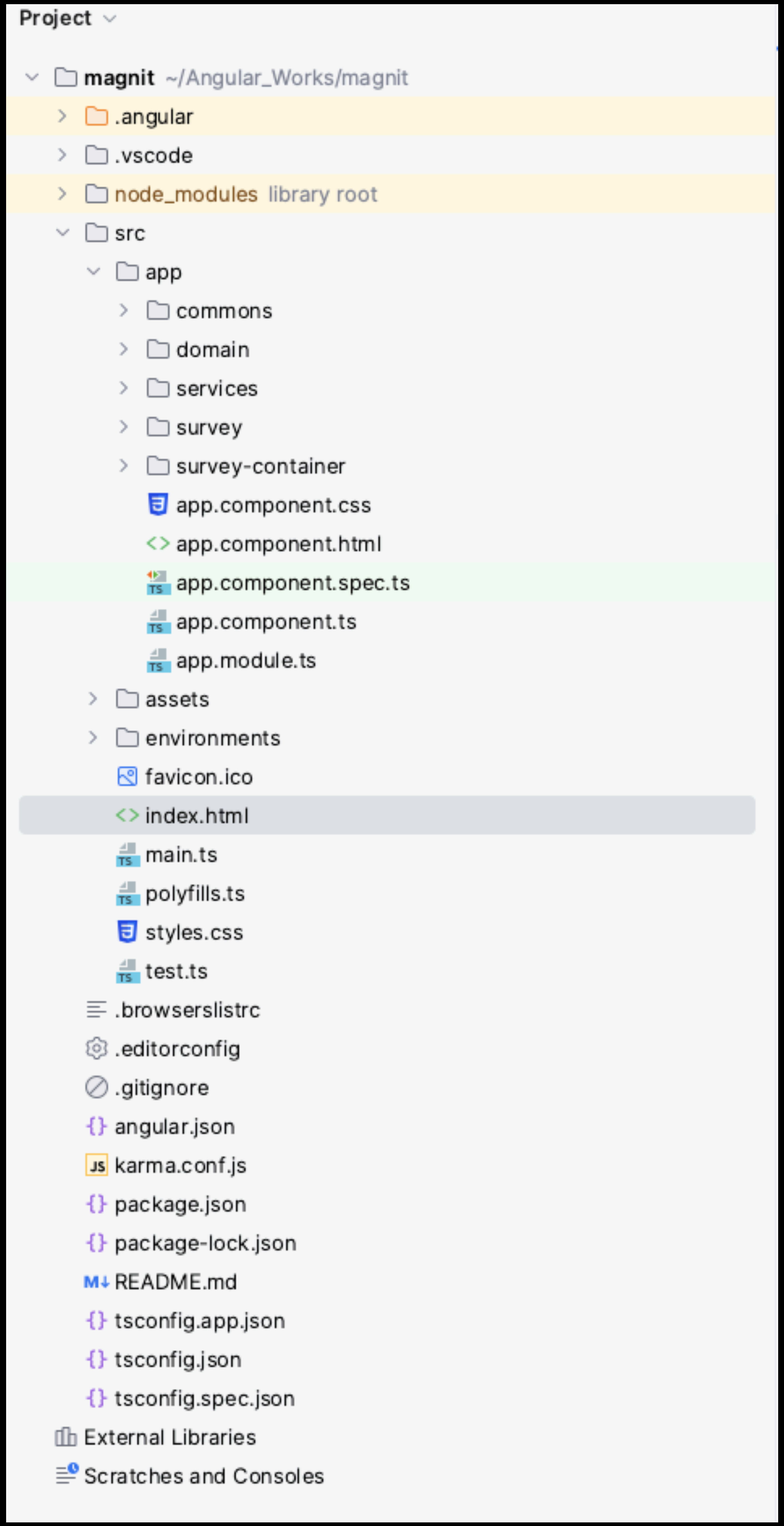
Magnit Hackathon

Employee Survey

Sainath Sanagapalli

UI - Angular

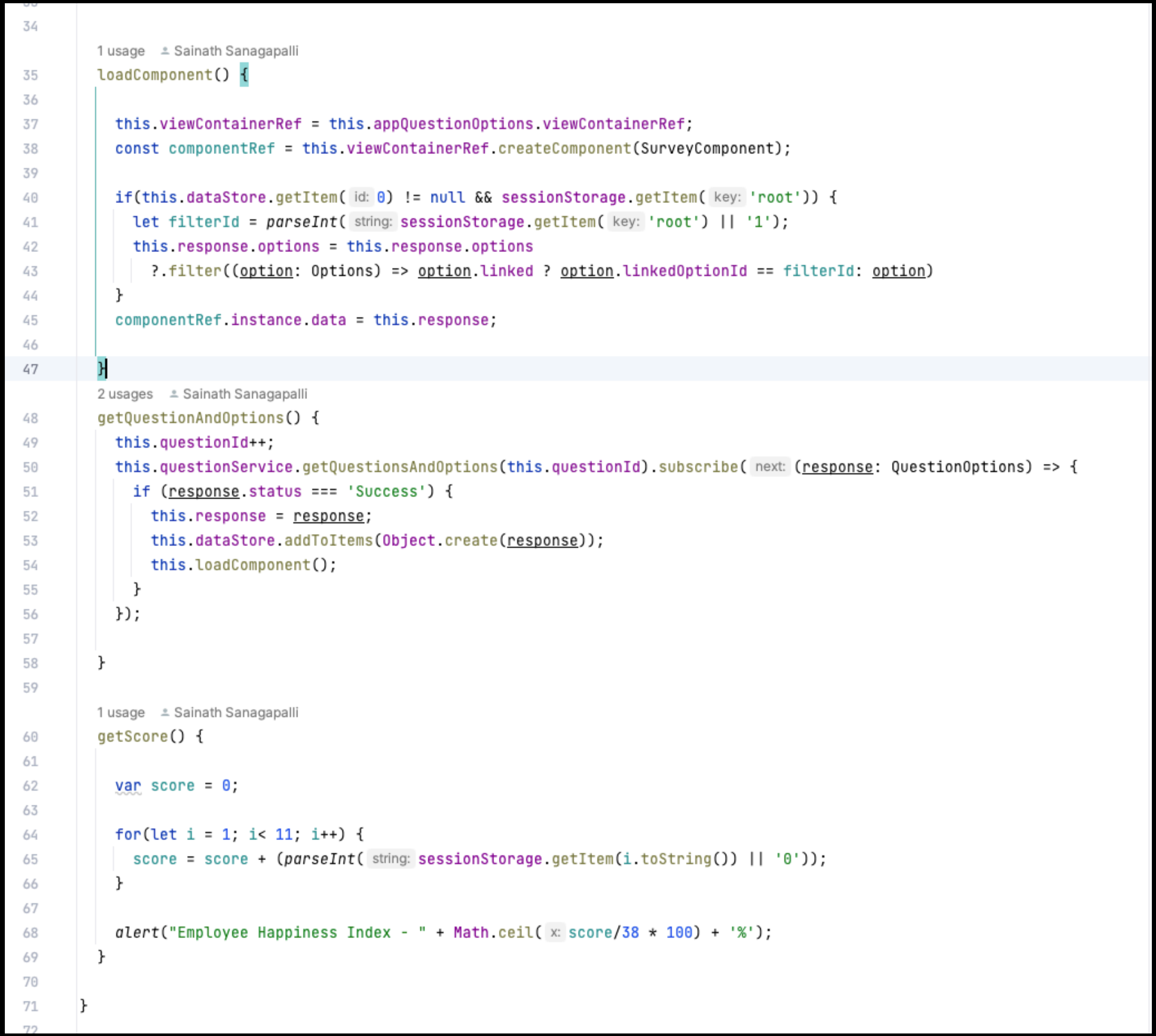
- Designed using Angular
- Commons - contains Directives for inserting components dynamically
- Domain - contains domain object for communication
- Services - code to call API
- Survey - common components that renders Questions and options
- SuveryContainer - Parent component for Survey Component



```
Project
magnit ~/Angular_Works/magnit
  .angular
  .vscode
  node_modules library root
  src
    app
      commons
      domain
      services
      survey
      survey-container
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
    assets
    environments
    favicon.ico
    index.html
    main.ts
    polyfills.ts
    styles.css
    test.ts
  .browserslistrc
  .editorconfig
  .gitignore
  angular.json
  karma.conf.js
  package.json
  package-lock.json
  README.md
  tsconfig.app.json
  tsconfig.json
  tsconfig.spec.json
External Libraries
Scratches and Consoles
```

UI - Core Explanation

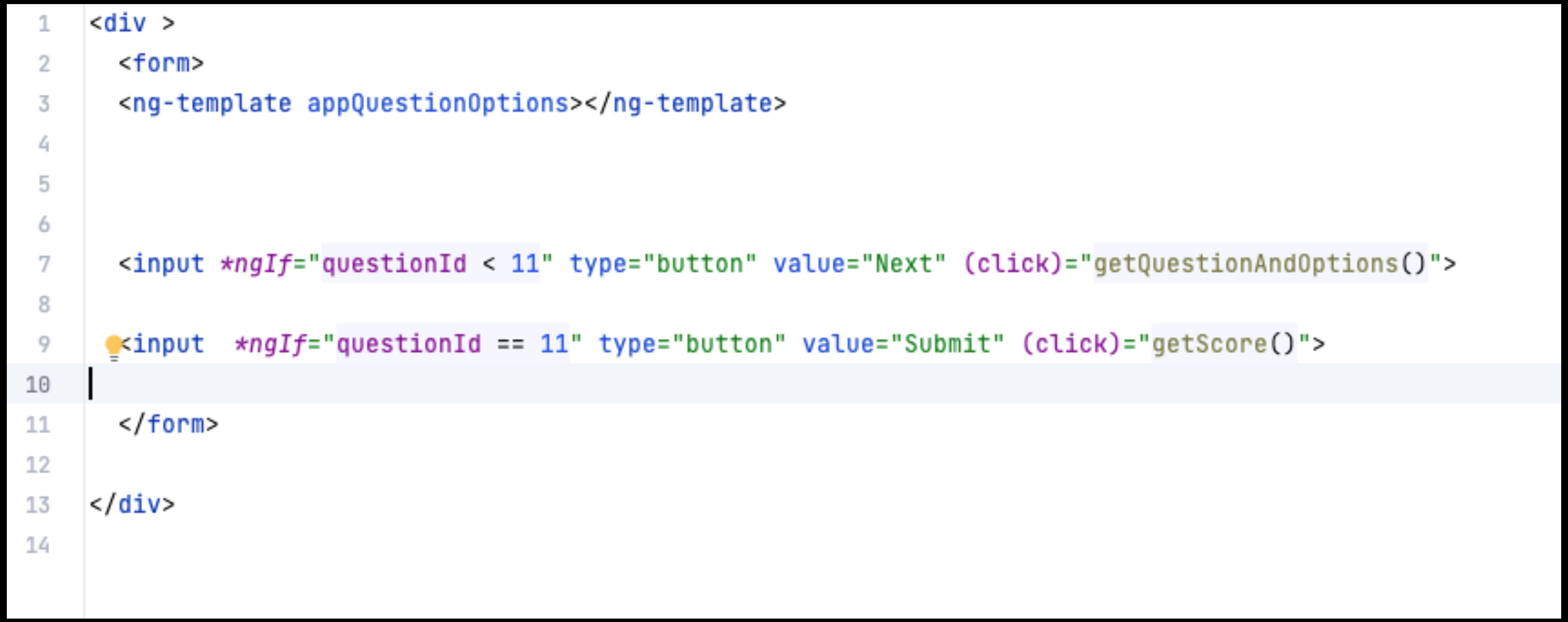
- Loads question using getQuestionAndOptions() that makes an API call to backend.
- loadComponent() inserts components dynamically into Survey Container
- getScore() - calculates happiness index.



```
  1 usage  Sainath Sanagapalli
  loadComponent() {

    this.viewContainerRef = this.appQuestionOptions.viewContainerRef;
    const componentRef = this.viewContainerRef.createComponent(SurveyComponent);

    if(this.dataStore.getItem( id: 0) != null && sessionStorage.getItem( key: 'root')) {
      let filterId = parseInt( string: sessionStorage.getItem( key: 'root') || '1');
      this.response.options = this.response.options
        ?.filter((option: Options) => option.linked ? option.linkedOptionId == filterId: option)
    }
    componentRef.instance.data = this.response;

  }
  2 usages  Sainath Sanagapalli
  getQuestionAndOptions() {
    this.questionId++;
    this.questionService.getQuestionsAndOptions(this.questionId).subscribe( next: (response: QuestionOptions) => {
      if (response.status === 'Success') {
        this.response = response;
        this.dataStore.addToItems(Object.create(response));
        this.loadComponent();
      }
    });

  }

  1 usage  Sainath Sanagapalli
  getScore() {

    var score = 0;

    for(let i = 1; i< 11; i++) {
      score = score + (parseInt( string: sessionStorage.getItem(i.toString()) || '0'));
    }

    alert("Employee Happiness Index - " + Math.ceil( x: score/38 * 100) + '%');
  }

}
```

Hook

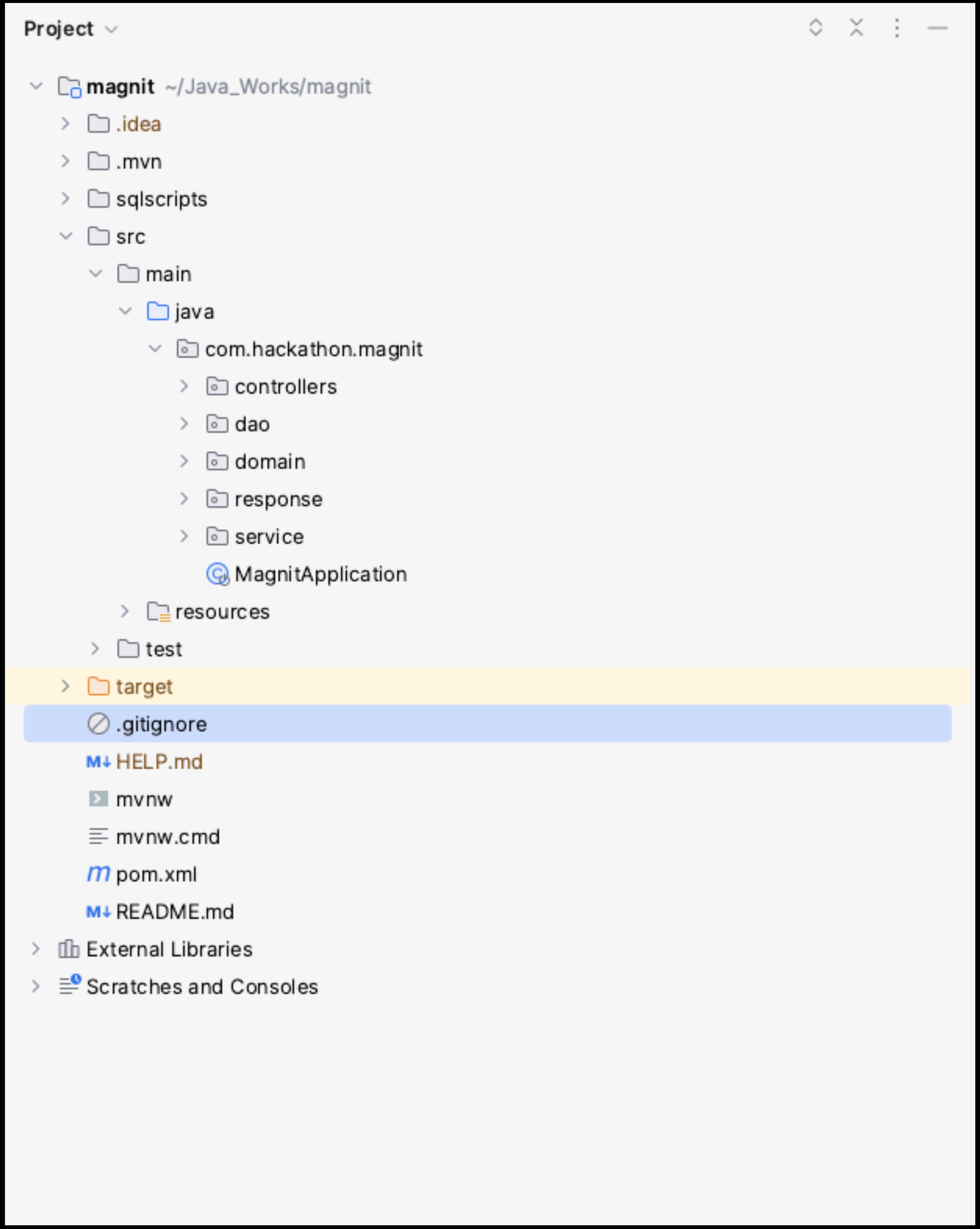
- <ng-template appQuestionOptions></ng-template> hooks dynamic components
- Keeps inserting components until Submit



```
<div >
  <form>
  <ng-template appQuestionOptions></ng-template>



  <input *ngIf="questionId < 11" type="button" value="Next" (click)="getQuestionAndOptions()">

  <input  *ngIf="questionId == 11" type="button" value="Submit" (click)="getScore()">

  </form>

</div>

```

API - SpringBoot

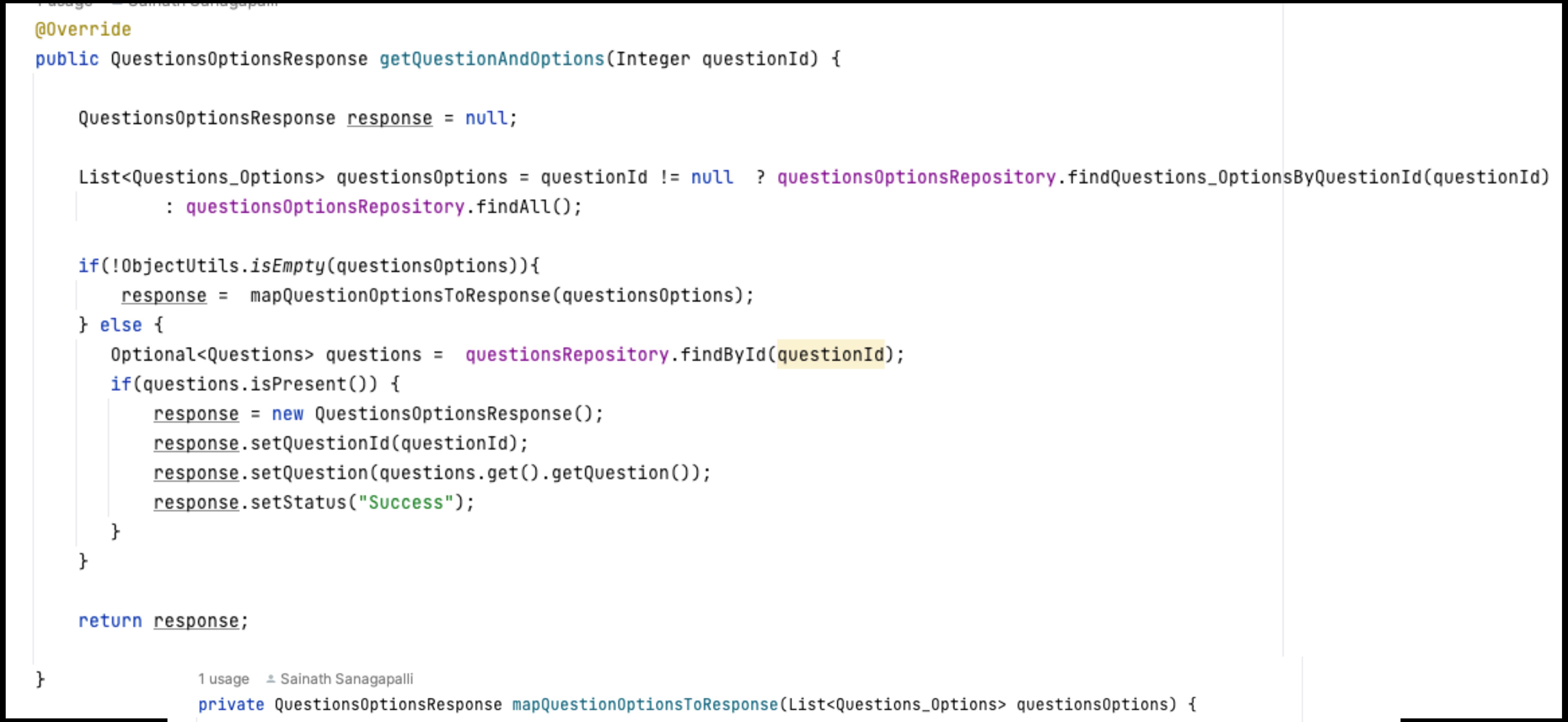
- Used SpringBoot and MySQL
- Controllers - API
- Dao - contains Repositories
- Domain - centaines entities
- Response - contains response objects
- Service - contains interface and implementation classes



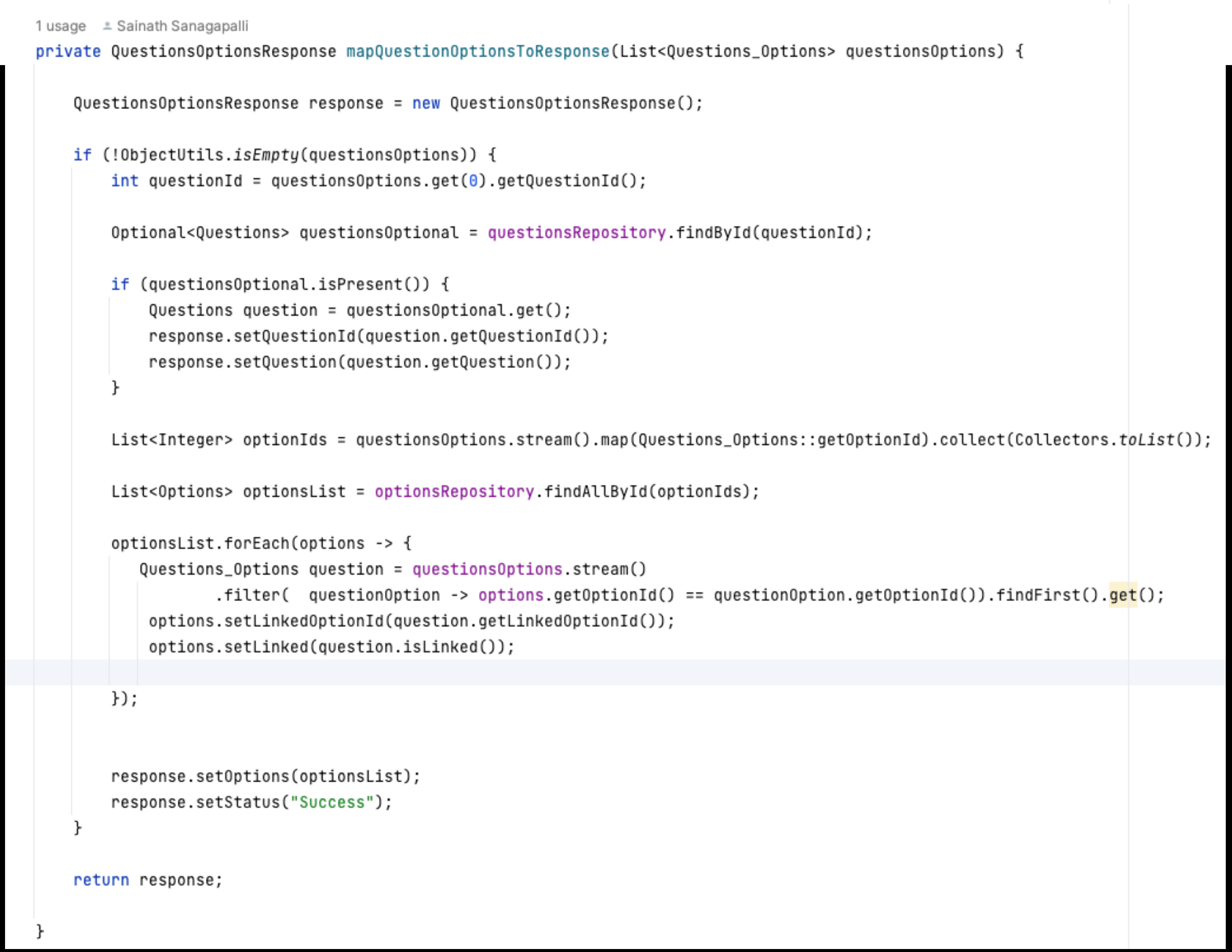
```
Project
magnit ~/Java_Works/magnit
  .idea
  .mvn
  sqlscripts
  src
    main
      java
        com.hackathon.magnit
          controllers
          dao
          domain
          response
          service
          MagnitApplication
      resources
    test
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
  README.md
External Libraries
Scratches and Consoles
```

API - Core Explanation

- Retrieves questions and options based on questionId passed
- Mapper class maps questions to options and linked options if any.



```
@Override
public QuestionsOptionsResponse getQuestionAndOptions(Integer questionId) {

    QuestionsOptionsResponse response = null;

    List<Questions_Options> questionsOptions = questionId != null  ? questionsOptionsRepository.findQuestions_OptionsByQuestionId(questionId)
            : questionsOptionsRepository.findAll();

    if(!ObjectUtils.isEmpty(questionsOptions)){
        response =  mapQuestionOptionsToResponse(questionsOptions);
    } else {
       Optional<Questions> questions =  questionsRepository.findById(questionId);
       if(questions.isPresent()) {
           response = new QuestionsOptionsResponse();
           response.setQuestionId(questionId);
           response.setQuestion(questions.get().getQuestion());
           response.setStatus("Success");
       }
    }

    return response;

}
```



1 usage Sainath Sanagapalli

```
private QuestionsOptionsResponse mapQuestionOptionsToResponse(List<Questions_Options> questionsOptions) {

    QuestionsOptionsResponse response = new QuestionsOptionsResponse();

    if (!ObjectUtils.isEmpty(questionsOptions)) {
        int questionId = questionsOptions.get(0).getQuestionId();

        Optional<Questions> questionsOptional = questionsRepository.findById(questionId);

        if (questionsOptional.isPresent()) {
            Questions question = questionsOptional.get();
            response.setQuestionId(question.getQuestionId());
            response.setQuestion(question.getQuestion());
        }

        List<Integer> optionIds = questionsOptions.stream().map(Questions_Options::getOptionId).collect(Collectors.toList());

        List<Options> optionsList = optionsRepository.findAllById(optionIds);

        optionsList.forEach(options -> {
           Questions_Options question = questionsOptions.stream()
                   .filter(  questionOption -> options.getOptionId() == questionOption.getOptionId()).findFirst().get();
            options.setLinkedOptionId(question.getLinkedOptionId());
            options.setLinked(question.isLinked());

        });


        response.setOptions(optionsList);
        response.setStatus("Success");
    }

    return response;

}
```

Thank you