AUTOMATIC QUESTION PAPER GENERATOR APPLICATION

Group Members:

Ramya Boyapati

Aparna Manda

Kishan Koushik Donthineni

Lohitha Yenugu

School of Computing & Engineering

University of Missouri - Kansas City

Contents:

Contributors:

This report contains entire documentation that is required for the project. Project was done by Ramya Boyapati, Aparna Manda, Kishan Koushik Donthineni, Lohitha Yenugu currently pursuing Web/Mobile Programming course.

Scope:

As Education being one of the most important sector, it requires technological stuff compared to other sectors. Question Paper Creation is a tedious and time consuming task. This system addresses this issue.

Objectives:

Faculty:

-Signup to the application and select the courses to generate question paper

-Navigate to Questions List, to add Descriptive/Multiple Choice Questions per course basis

-Navigate to Student List, to view the list of students enrolled into courses

-Go to courses, to generate question paper

Student:

-Sign up to the application and enroll into courses

-Navigate to Enrollment to view the courses enrolled

Methodology:

Phase 1: Database Design and User Registration

Created database objects in Mongo DB.

1. CourseSchema

2. DescriptiveQuestionsSchema
3. FacultySchema
4. FacultyCourseSchema
5. FacultyCourseQuestionsSchema
6. FacultyQuestionPaperSchema
7. MultipleQuestionsSchema
8. StudentSchema
9. StudentAnswersSheetSchema
10. StudentCourseSchema

Created Student and Faculty, Login and Signup pages.

Phase 2: Faculty Module

Created pages

1. My Profile
2. Courses
3. Student List
4. Questions List
5. Add Questions
6. Generate Paper

Phase 3: Student Module

Created pages

1. My Profile
2. Enrollments
3. Assignments

Software and Libraries:

MEAN Stack

Mongo DB

Node.js

Express.js

Code Snippets:

```
{
  this.api.getFacultyCourses()
      .subscribe( next: data => {
        //console.log(data);
        this.enrollments = data;
        for (let i = 0; i < this.enrollments.length; i++) {
          this.enrollments[i].ind = i;
          this.api.getCourse(this.enrollments[i].cid)
              .subscribe( next: res => {
                console.log(res)
                    this.enrollments[i].CourseName = res.CourseName;
console.log(this.enrollments[i].CourseName);
                //console.log(this.enrollments[i]);
                  }, error: (err) => {
                    console.log(err);
                  }
              );
          //console.log(this.enrollments[i]);
          this.api.getFaculty(this.enrollments[i].fid)
              .subscribe( next: res => {
                  // console.log(res)
                    this.enrollments[i].FacultyName = res.name;

                  //  console.log(this.enrollments[i]);
                  }, error: (err) => {
                    console.log(err);
                  }
              );
          this.api.getStudentCourseExists(this.enrollments[i]._id,this.user_id).subscribe( next: res => {
                //console.log(res)
                if(res.length>0)
```

The above code is written to list the available courses for the logged in student and also lists the courses the student is already enrolled into.

```
        question['pts'] = $('#Pt').val();
        this.api.postDescriptiveQuestions(question)
            .subscribe( next: res => {
                let questionfc: object = {};
                questionfc['qid'] = res._id;
                questionfc['dmq'] = 'D';
                questionfc['fcid'] = this.fc_id;
                this.api.postFacultyCourseQuestions(questionfc)
                    .subscribe( next: res => {
                        console.log("****")
                        console.log(res);
                        console.log(this.fc_id)
                        this.dialogRef.close();

                        }, error: (err) => {
                            // console.log(err);
                        }
                    );

                }, error: (err) => {
                    // console.log(err);
                }
            );
    }
    else {
        question['OptA'] = $('#A').val();
        question['OptB'] = $('#B').val();
        question['OptC'] = $('#C').val();
        question['OptD'] = $('#D').val();
        question['AnsOpt'] = $('#opt').children("option:selected").val();
        this.api.postMultipleQuestions(question)
```

The above code is written to add Descriptive Questions and Multiple Choice questions to the respective database tables. It first retrieves data from html elements and is posted to the tables.

```
let id_2="#A"+m;
let dlevel = $(id_1).children("option:selected").val();
let num_elems=$(id_2).val();
console.log(dlevel);
this.api.getFacultyQuestionsLevel(this.fc_id,dlevel, fct: 'D')
    .subscribe( next: res => {
            let max=0;
            let k2=0;
            max=res.length-1;
            while(k2<num_elems)
            {
                let rn= Math.floor( x: Math.random() * max) ;
                console.log(res);
                console.log(rn)
                this.res5[k2]=res[rn].qid;
                k2++;

            }

            if(m==((this.i)))
            {
                console.log(this.res5);
                for(let g=0;g<this.res5.length;g++)
                {
                    this.api.getDescriptive(this.res5[g])
                        .subscribe( next: res => {
                            res['ind']=g+1;
                                this.final_res1[this.fr]=res;
                                if(g==((this.res5.length-1)))
                                {
```

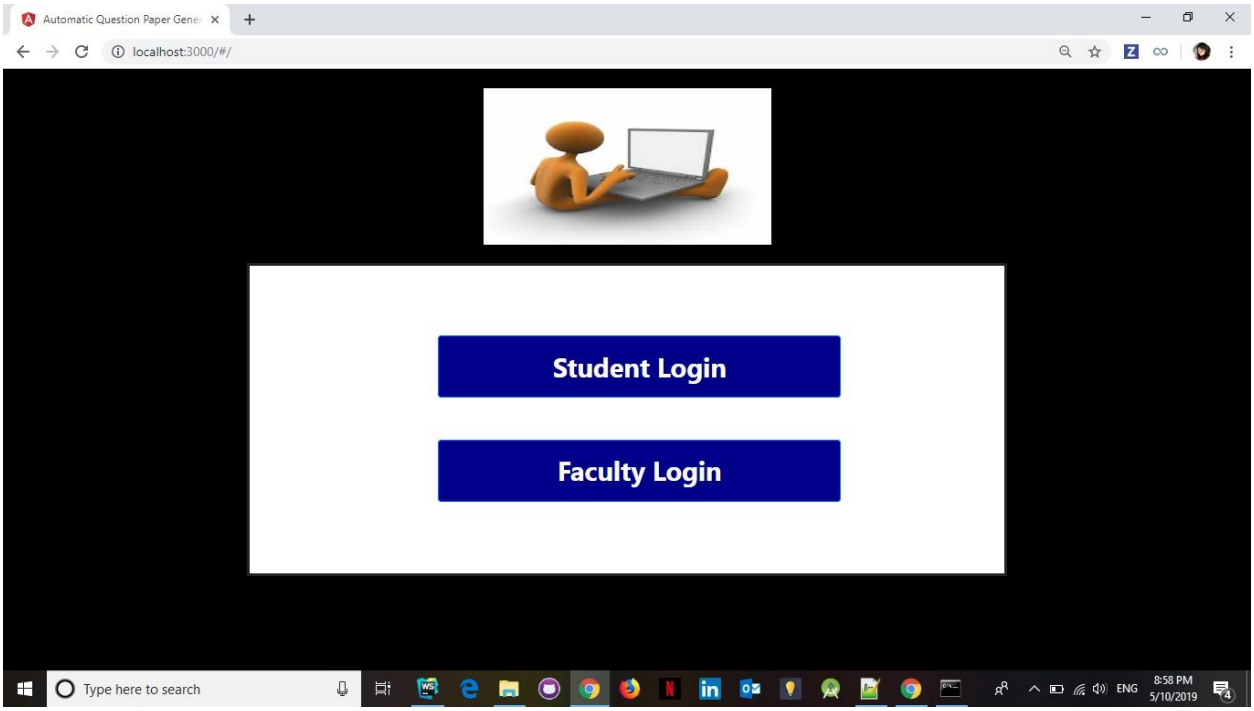
The above code fetches all the records for the selected course and for the given difficulty level. It then runs random function on these records to generate questions. This is repeated for different levels of difficulty.

```
    let f = document.getElementsByClassName( classNames: "ro");
    for(let i=0,fLen=f.length;i<fLen;i++){
        f[i].removeAttribute( qualifiedName: "readonly");
    }
    f = document.getElementsByClassName( classNames: "dis");
    for(let i=0,fLen=f.length;i<fLen;i++){
        f[i].removeAttribute( qualifiedName: "disabled");
    }
    document.getElementById( elementId: "save").style.display="";
    document.getElementById( elementId: "edit").style.display="none";
}
onCollapse()
{
    if(this.collapse==true)
    {this.collapse=false;
    document.getElementById( elementId: "content-div").classList.add("col-12");
        document.getElementById( elementId: "content-div").classList.remove( token: "col-10");}
    else
    {this.collapse=true;
        document.getElementById( elementId: "content-div").classList.add("col-10");
        document.getElementById( elementId: "content-div").classList.remove( token: "col-12");}
}
onSave()
{
    this.submitted = true;
    if (this.userForm.invalid) {
        return;
    }
    let user: object = {};
    user['userID'] = 18;
    user['name'] = this.userForm.value.Name;
```

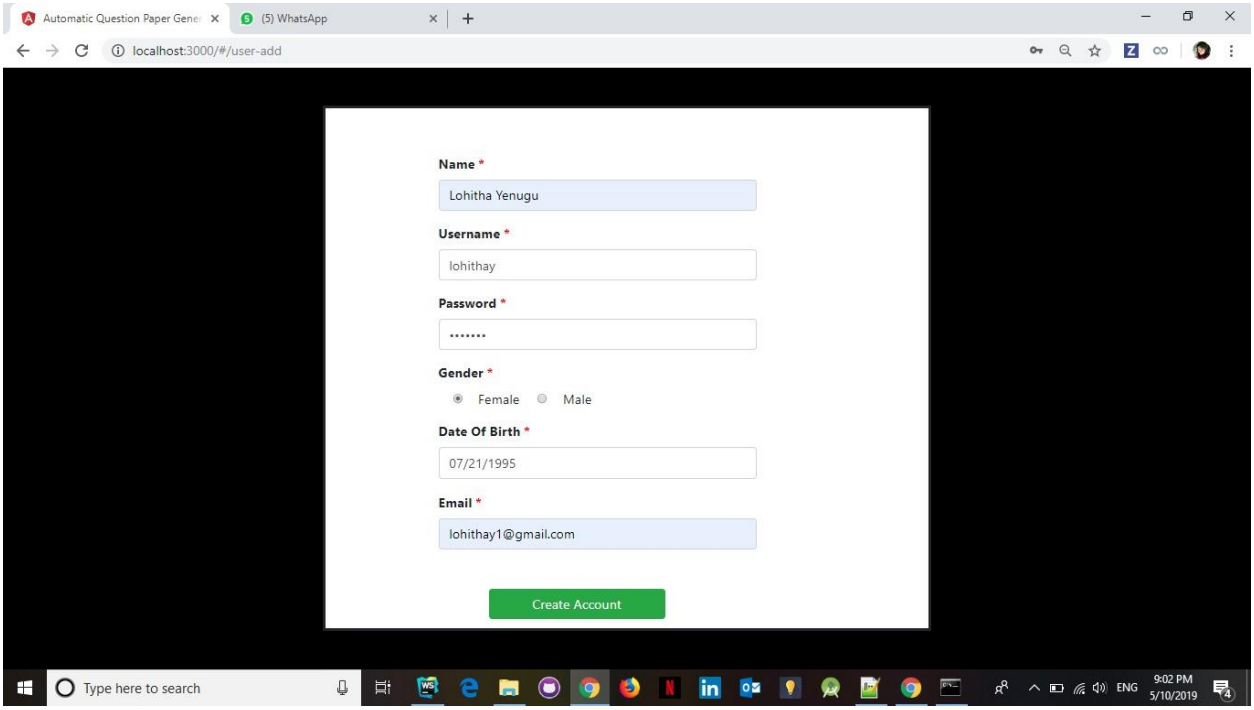
The above code is used to hide the navigation bar and to update the profile of the user.

Implementation:

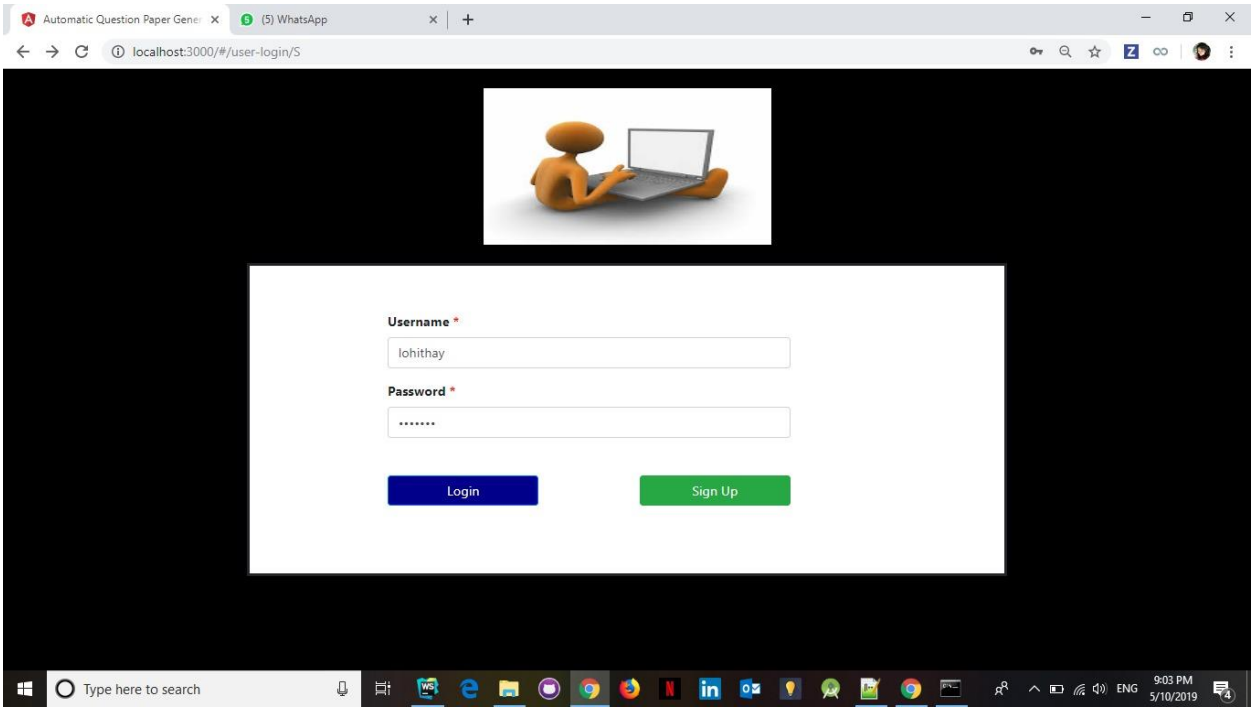
- This is the User Registration page



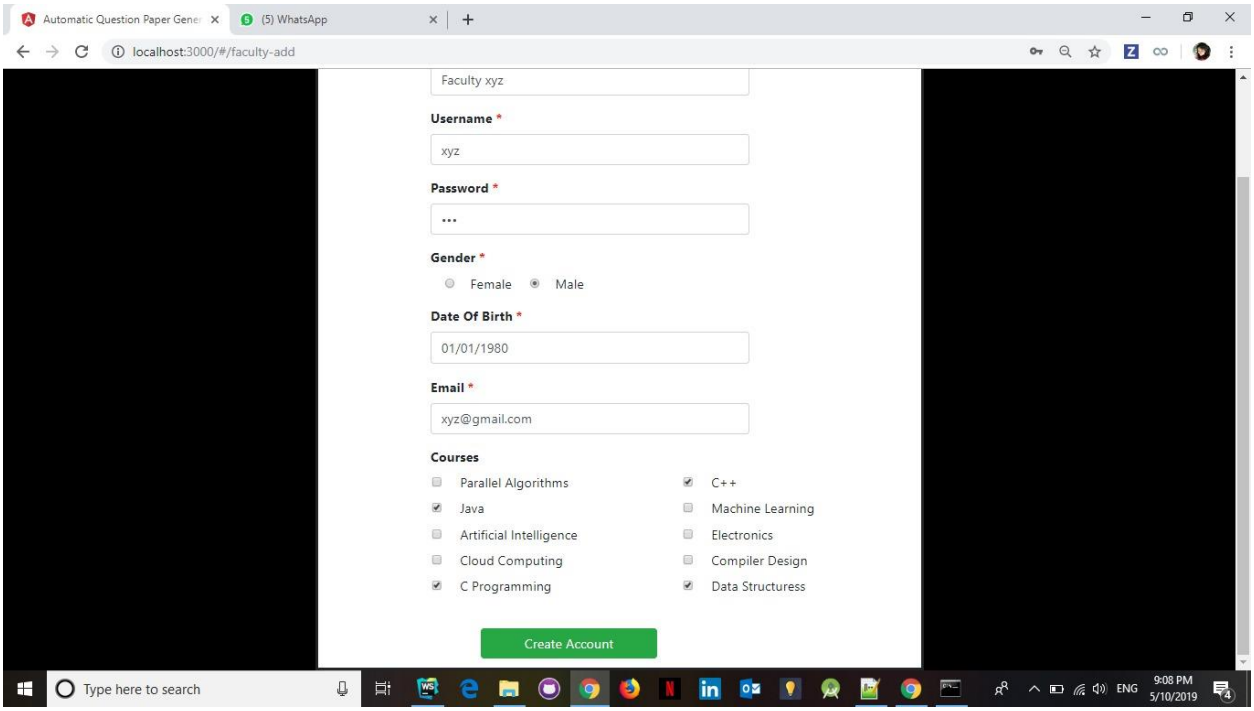
- On clicking Student Login, Student signs up using this page



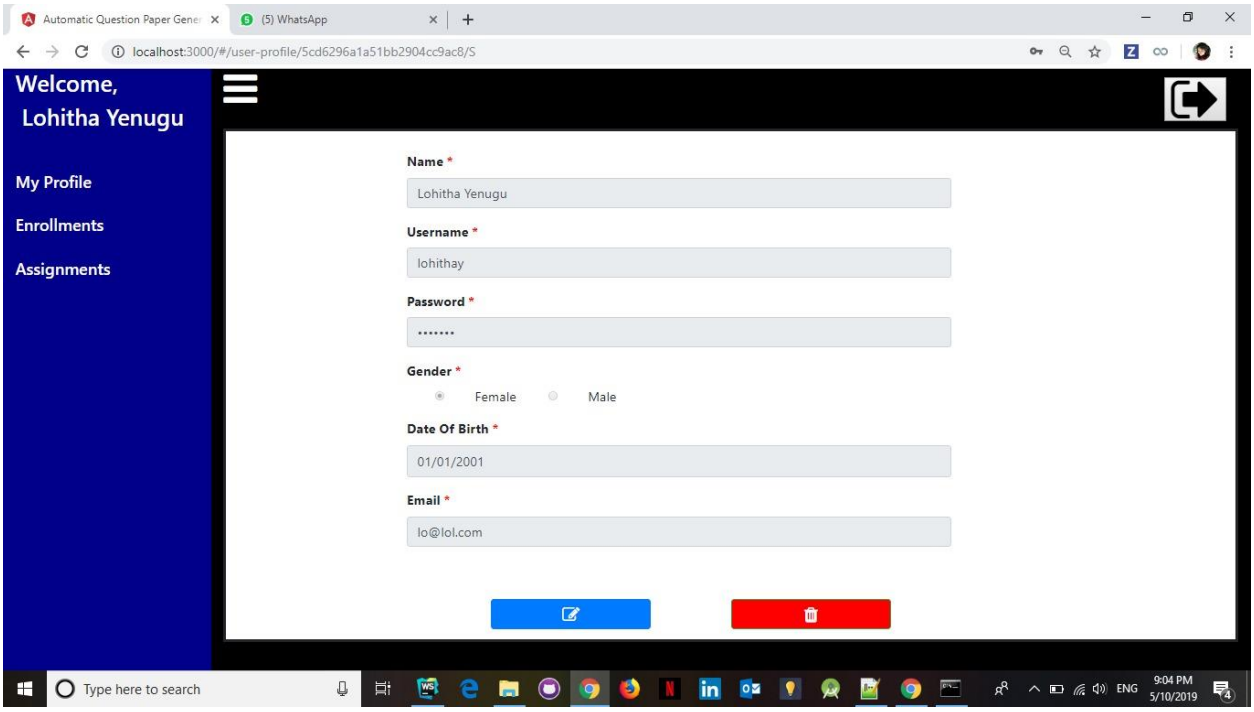
- Now, Student can login into the application with his credentials



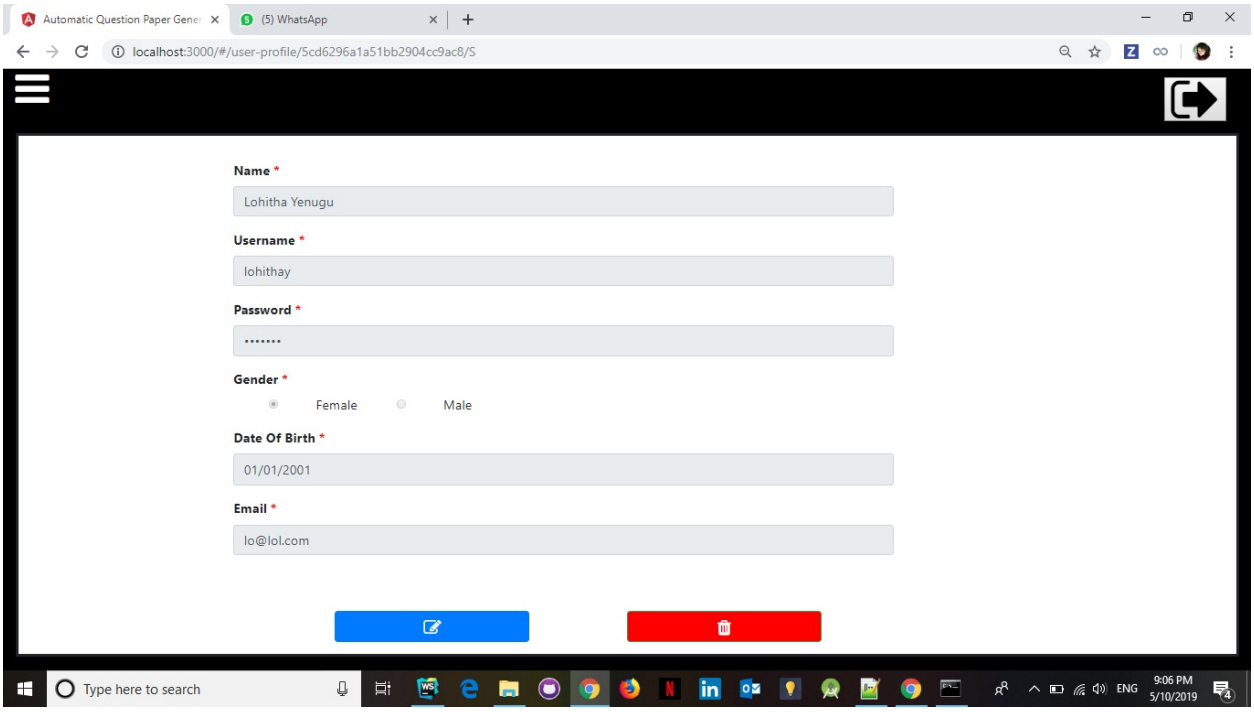
- On clicking Faculty Login, Faculty signs up using this page and selects list of courses to generate question paper from the available list



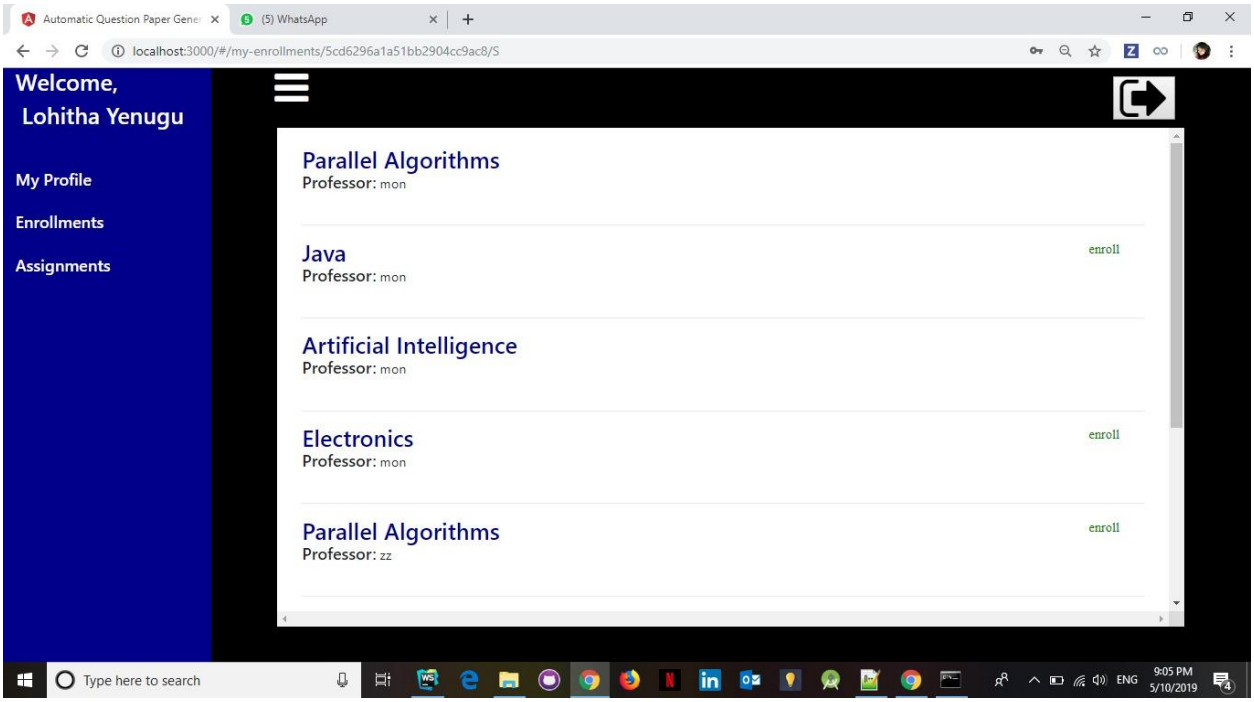
- This is the Student's Profile page where student can update his details or delete his record.



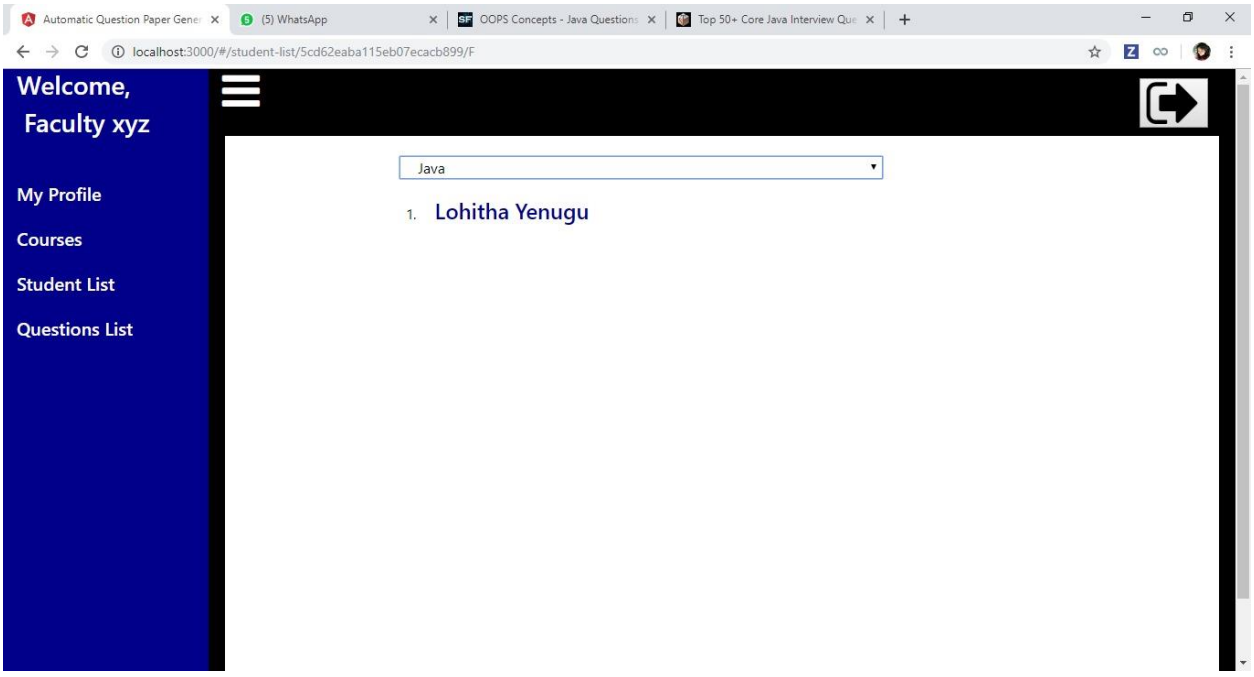
- Navigation bar is collapsible allowing to view the details in full page mode



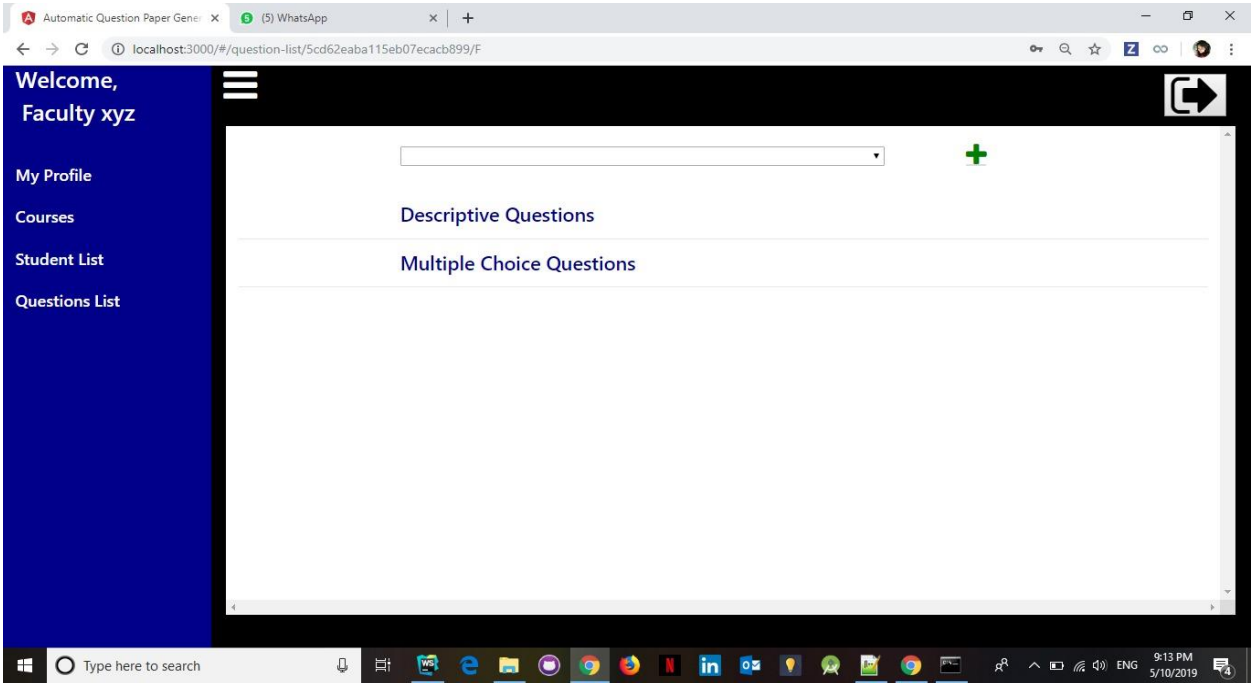
- Enrollments page displays list of courses available and student can enroll to the courses



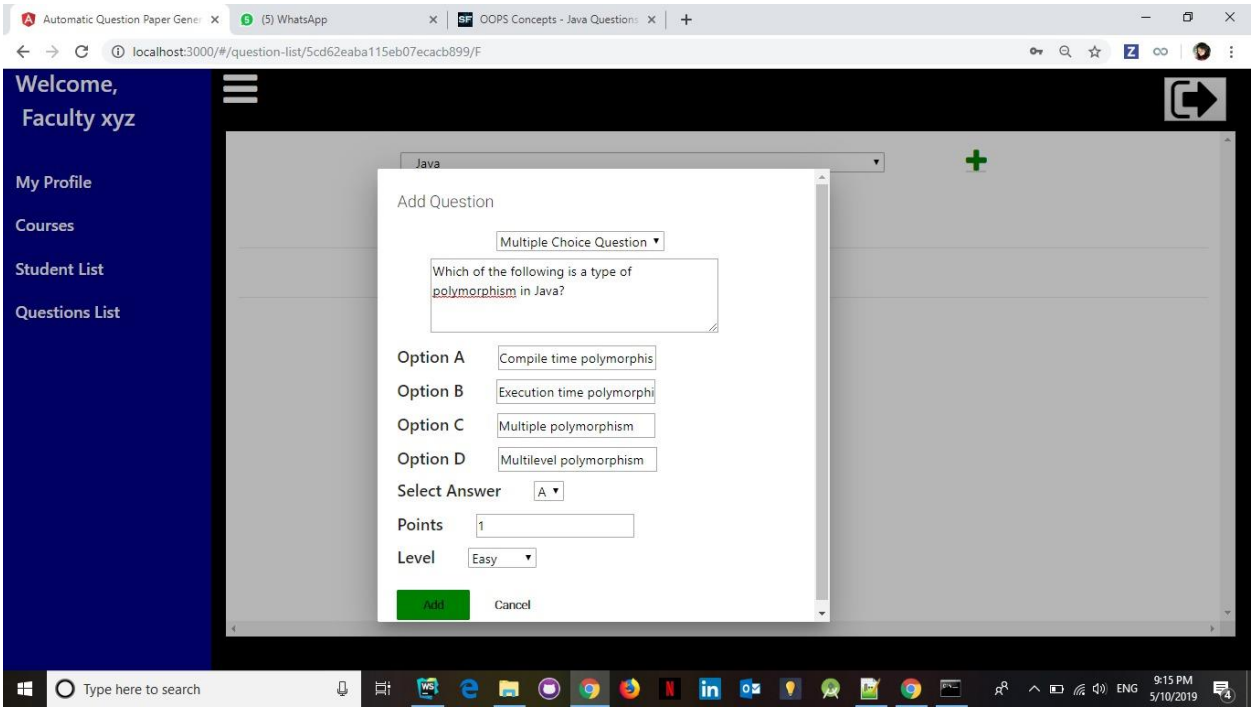
- Student List page shows the list of students enrolled into the selected course offered by logged in Faculty



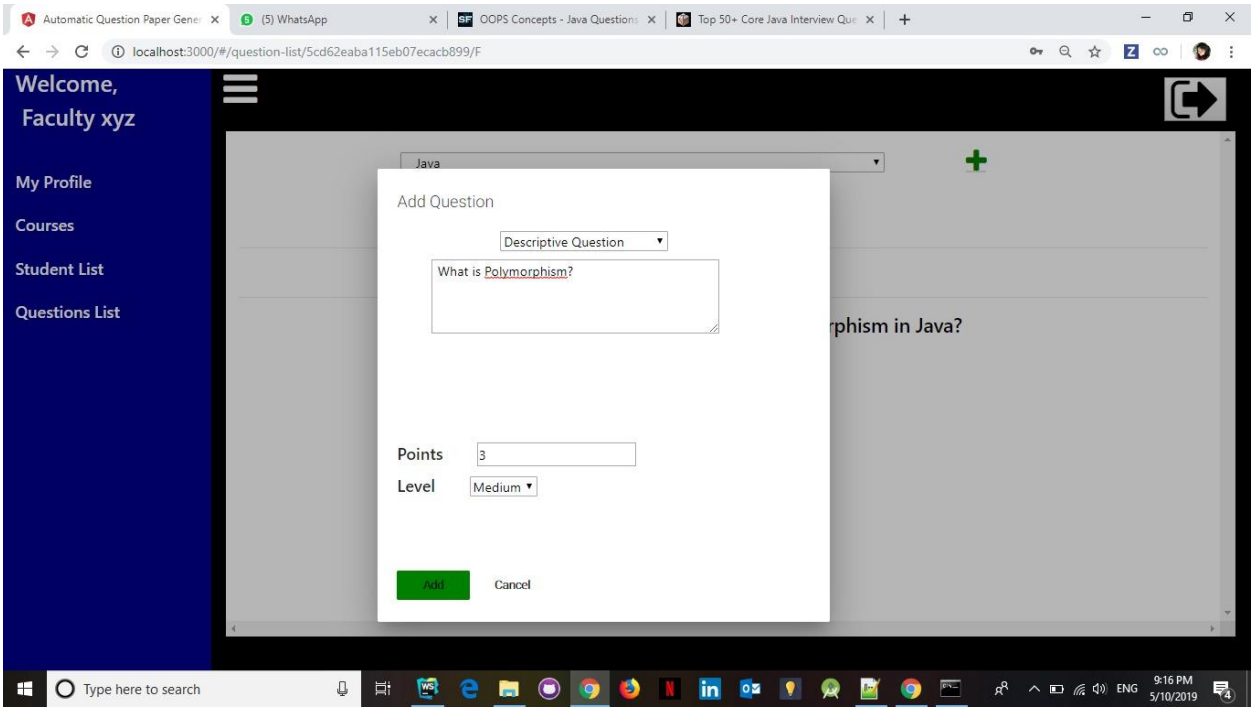
- Questions List page lists all the questions (Descriptive and Multiple Choice) added by the Faculty for the selected course. The '+' button allows the Faculty to add questions.



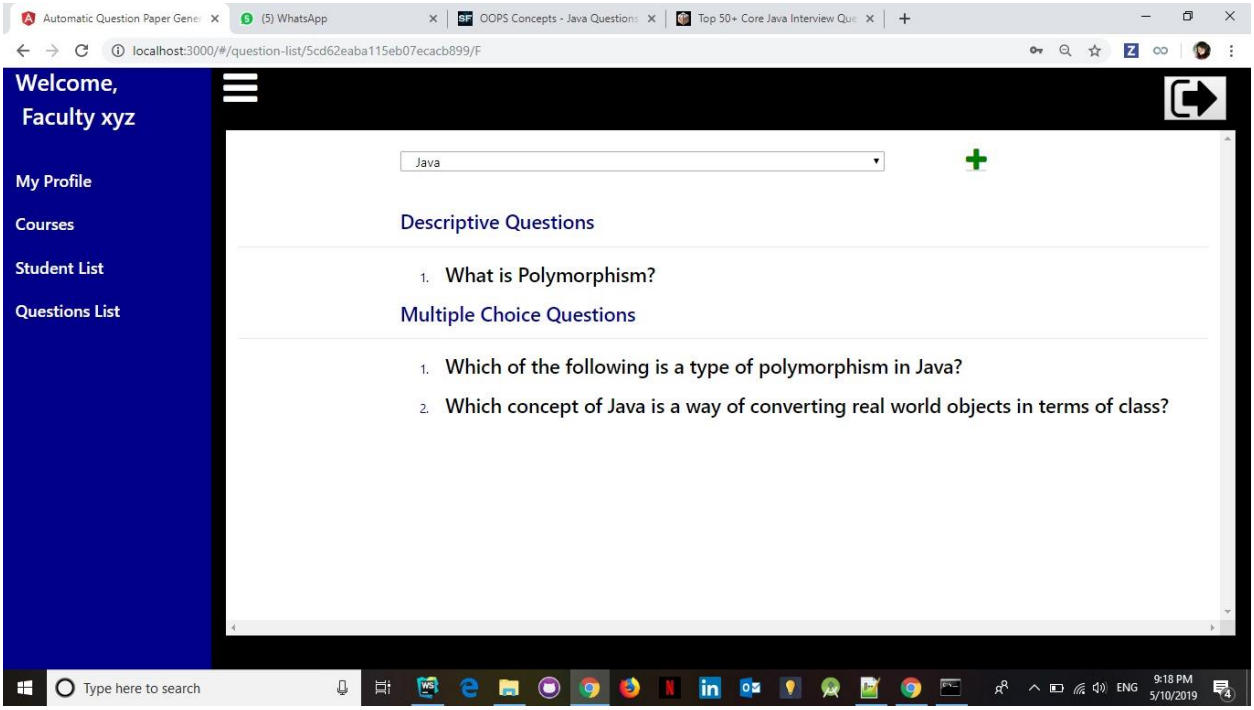
- Model window to add Multiple Choice Question



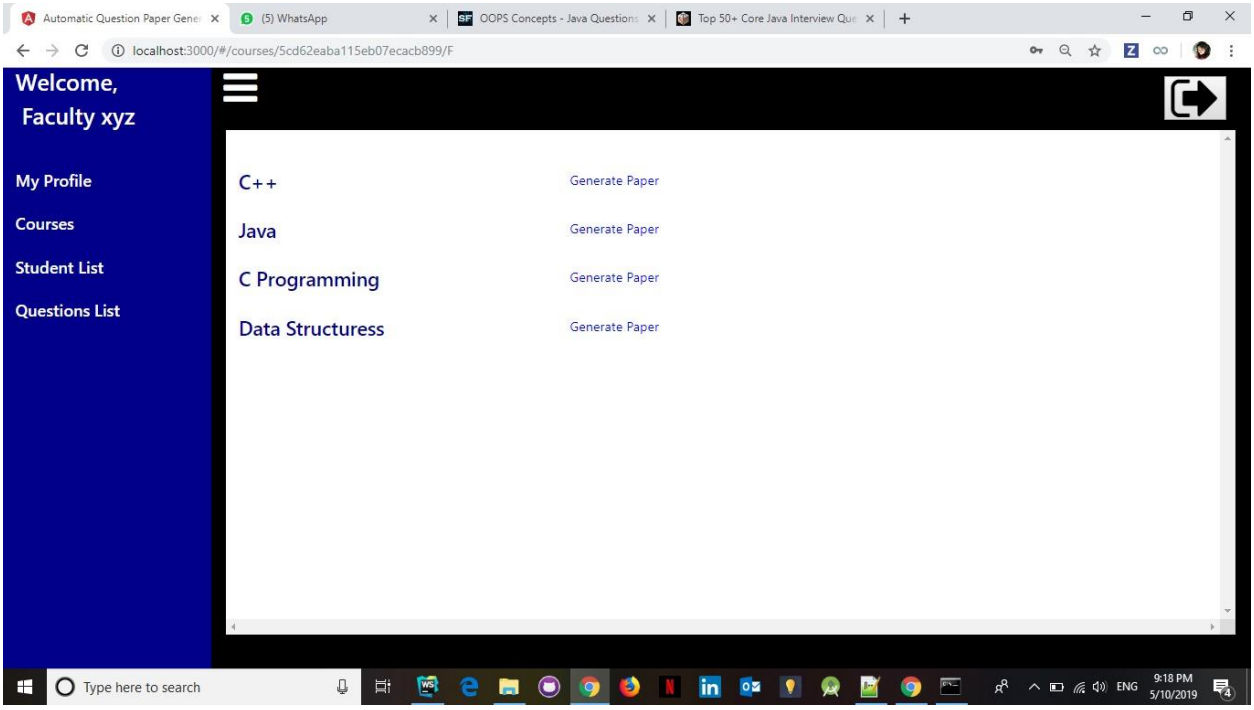
- Model window to add Descriptive Question



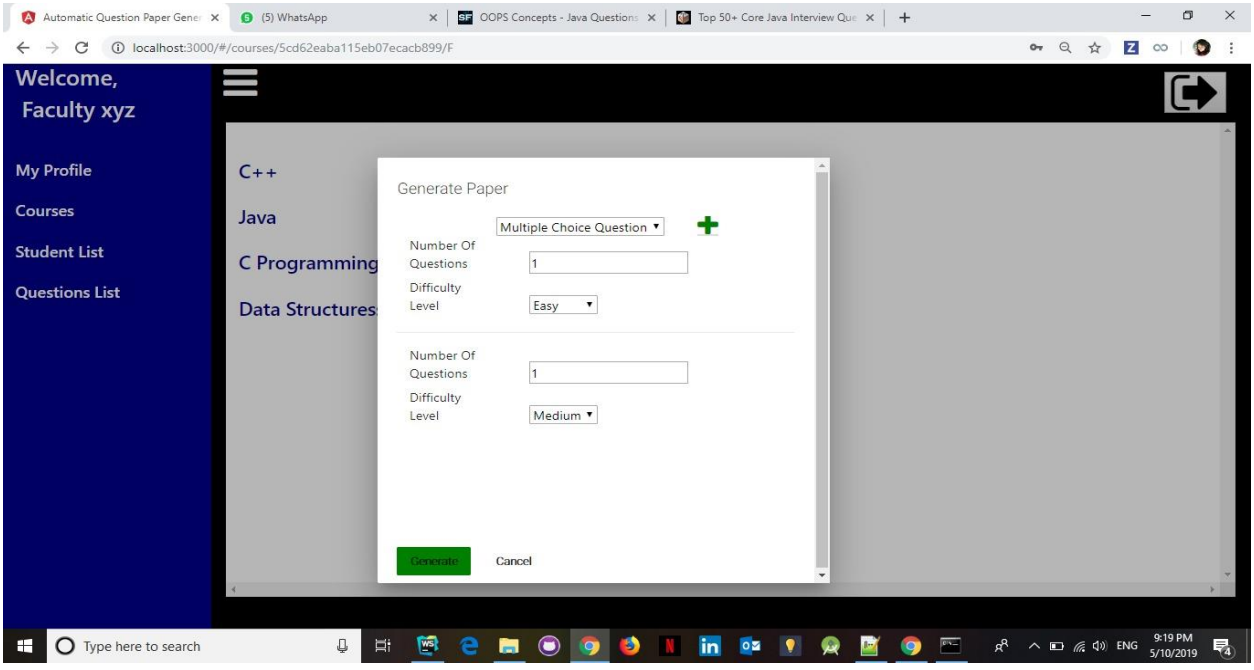
- Here, the added questions are listed for the selected course



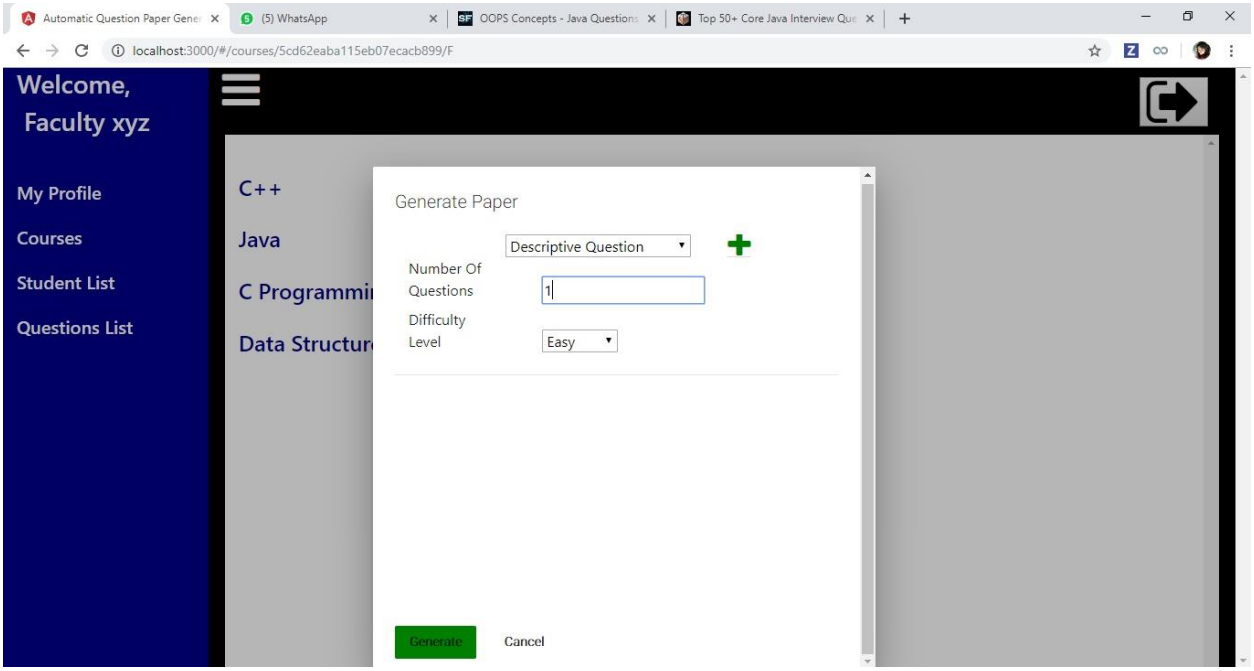
- Courses page shows list of courses the faculty signed for and allows to generate Question Paper for the respective course. Model window shows up on clicking ‘Generate Paper’.



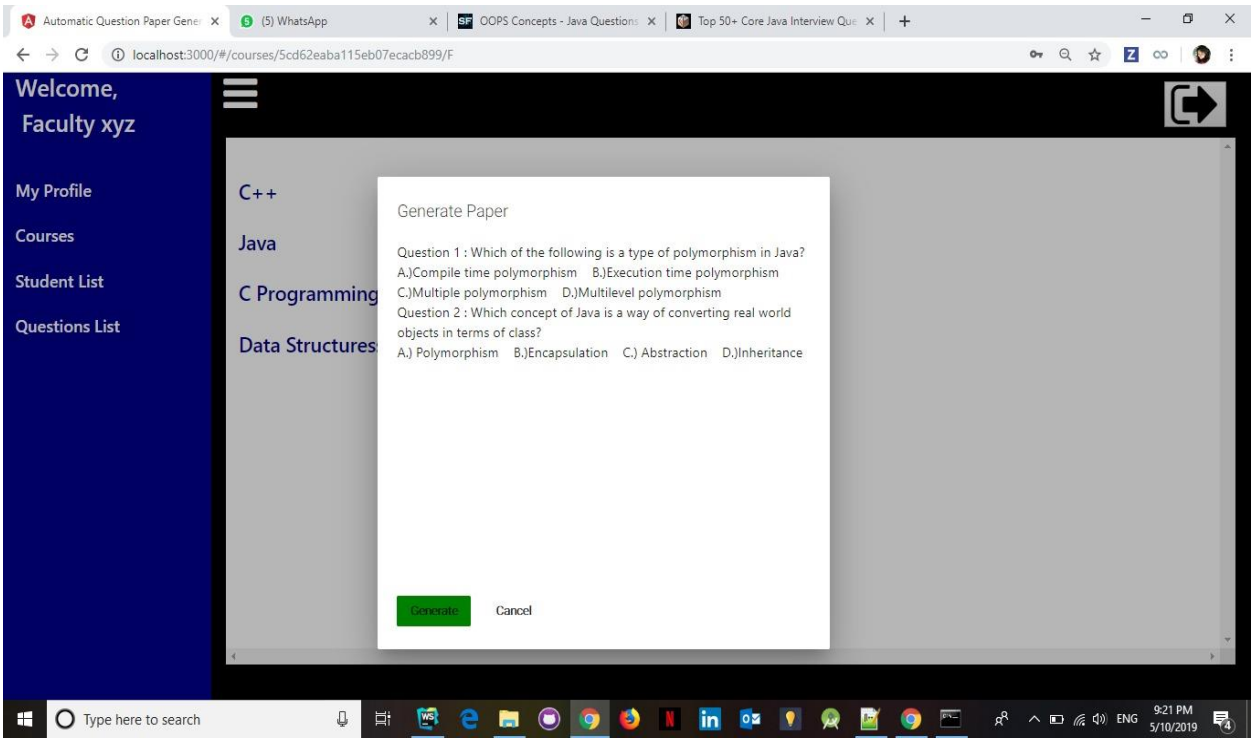
- Model window to generate paper with Multiple Choice questions



- Modal window to generate paper with Descriptive Questions



- This is the generated paper for the given input



Conclusion:

In this project, an automated Question Paper is generated which is implemented using MEAN stack technology. We have also considered the importance of randomization in the task of paper generation. Our project uses randomized method to generate questions, making it impossible to derive any pattern in the papers. We distinguished between faculty and students by their tasks.

Links:

Github Link: https://github.com/mandaaparna

References:

https://stackoverflow.com/questions/3087975/make-the-cursor-a-hand-when-a-user-hovers-over-a-list-item

https://blog.angular-university.io/angular-material-dialog/

https://stackoverflow.com/questions/28175381/how-to-access-json-array-object-in-html-using-angular

https://www.w3schools.com/angular/ng_ng-repeat.asp

https://stackoverflow.com/questions/43289366/angular2-routing-the-requested-path-contains-undefined-segment-at-index-1