UNIVERSITY of GREENWICH
Alliance with FPT Education

BTEC

Higher Nationals in Computing

Unit 30: Application Development

ASSIGNMENT 2

Learner's name: Huynh Cam Hung

ID: GCS18026

Class: GCS0701A

Subject code: 1670

Assessor name: **LE NGOC THANH**

Assignment due: 10 May 2020

Assignment submitted: 10 May 2020

UNIVERSITY of GREENWICH
Alliance with FPT Education

BTEC

ASSIGNMENT 2 FRONT SHEET

Qualification	BTEC HND Diploma in Computing and Systems Development		
Unit number and title	Unit 30: Application Development		
Assignment due	10 May 2020	**Assignment submitted**	10 May 2020
Learner's name	Huynh Cam Hung	**Assessor name**	Le Ngoc Thanh

Learner declaration: I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.			
Learner signature		**Date**	

Grading grid

P3	P4	P5	M3	M4	D3	D4

Assignment title	**Assignment 2: Application Implementation and Evaluation**

In this assignment, you will have opportunities to provide evidence against the following criteria. Indicate the page numbers where the evidence can be found.

Assessment criteria	Expected evidence	Task	Assessor's Feedback
LO3: Work individually and as part of a team to plan and produce a functional business application with support documentation			
P4 Create a formal questionnaire that effectively reviews your business application, problem definition statement, proposed solution and development strategy. Use this questionnaire as part of a peer-review and document any feedback given.	1. Questionnaires to peer-review your application	1	
P5 Develop a functional business application based on a specified business problem	2. A runnable implementation of the application		
LO4: Evaluate the performance of a business application against its Software Design Document and initial requirements			

P4. Review the performance of your business application against the Problem Definition Statement and initial requirements	3. Test plan 4. Test results	3	

Assessment criteria	**Expected Evidence**	**Feedback** **(note on Merit/Distinction if applicable)**
Merit descriptor No. (M3)		
Merit descriptor No. (M4)		
Merit descriptor No. (M5)		
Distinction descriptor No. (D2)		
Distinction descriptor No. (D3)		
Summative feedback		

Assessor's Signature		Date	

Unit Number and Title	**30: Application Development**
Academic Year	2018
Unit Tutor	To Hoai Viet
Assignment Title	**Assignment 2: Application Implementation and Evaluation**
Issue Date	
Submission Date	10 May 2020
IV Name & Date	

Learning Outcomes and Assessment Criteria		
Pass	**Merit**	**Distinction**
LO3 Work individually and as part of a team to plan and produce a functional business application with support documentation		
P4 Create a formal questionnaire that effectively reviews your business application, problem definition statement, proposed solution and development strategy. Use this questionnaire as part of a peer-review and document any feedback given. **P5** Develop a functional business application based on a specified business	**M3** Interpret your peer-review feedback and identify opportunities not previously considered. **M4** Develop a functional business application based on a specific Software Design Document with supportive evidence of using the preferred tools, techniques and methodologies.	**D2** Evaluate any new insights, ideas or potential improvements to your system and justify the reasons why you have chosen to include (or not to include) them as part of this business application.

problem.		
LO4 Evaluate the performance of a business application against its Software Design Document and initial requirements		
P6 Review the performance of your business application against the Problem Definition Statement and initial requirements.	**M5** Analyse the factors that influence the performance of a business application and use them to undertake a critical review of the design, development and testing stages of your application. Conclude your review by reflectively discussing your previously identified risks.	**D3** Critically evaluate the strengths and weaknesses of your business application and fully justify opportunities for improvement and further development.

Assignment Brief and Guidance
As the technology is being developed rapidly nowadays, FPT Co. desires to build the continuing study environment throughout the corporation. It is necessary to develop a web-based system, which manages the activity of "Training" for internal training program of the company. This system can be used to manage trainee accounts, manage trainers, manage course categories, manage courses, manage topics, assign topic to course, assign trainer to topic, assign trainee to course. This is a system used by HR department. We have three roles in this system, an administrator, training staff and a trainer. The brief description of those roles is as follow. 1. An administrator's role ● Can login to the system through the first page of the application ● Can create/edit/delete new user account for trainer/training staff and assign/change(if existing user) username and a password

2. A training staff's role
    - A registered training staff, who is assigned a user name and a password by the administrator logs in can create trainee accounts by entering details like trainee name, trainee accounts, age, date of birth, education, main programming language, TOEIC score, experience details, department, location, etc.
    - After entering successfully all details for trainees, his/her details are then stored in the database. The training staff is given a list of trainees for him to view and search. From the list of trainees, he can also search by trainee account, programming language, TOEIC score…
    - Can update, delete trainee accounts
    - Can manage course categories such as searching, adding, updating and deleting course categories. Course category includes the information such as course category name and descriptions.
    - Can manage courses such as searching, adding, updating and deleting courses. Course includes course name and description.
    - Can add topics such as topic name and topic descriptions into a course, add courses into a category.
    - Can manage trainer profile such as adding, updating and deleting the information: Trainer name, External or Internal Type, working place, telephone, and email address.
    - Can assign trainer to a topic.
    - Can assign trainee to a course.
3. A trainer's role
    - In the same system, the trainer who have been registered by the administrator can login and can update his profile such as Trainer name, External or Internal Type, education, working place, telephone, and email address.
    - Can view courses which have a topic he is assigned to.

Your manager suggests that this would be a great opportunity for you to demonstrate your capabilities by designing and developing the application. After considering, you decide to do the project. The project consists of 4 steps which is divided into two phases. After two first steps of analysing and designing the solution, now it is time to implement and evaluate the application.		
Assignment Guidance		
Task	**Assessment Criteria**	**Requirement**
1	**P4** Create a formal questionnaire that effectively reviews your business application, problem definition statement, proposed solution and development strategy. Use this questionnaire as part of a peer-review and document any feedback given.	1. Questionnaires to peer-review your application
2	**P5** Develop a functional business application based on a specified business problem.	2. A runnable implementation of the application
3	**P6** Review the performance of your business application against the Problem Definition Statement and initial requirements.	3. Test plan 4. Test results
4	**M3** Interpret your peer-review feedback and identify opportunities not previously considered.	5. Peer-review feedback and Evaluation on feedback 6. Recommendation for improvements
5	**M4** Develop a functional business application based on a specific Software Design Document with supportive evidence of using the preferred tools, techniques and methodologies.	7. Sequence diagrams to show business process of important application functions
6	**M5** Analyse the factors that influence the performance of a business application and use them to undertake a critical review of the design, development and testing stages of your application. Conclude your review by reflectively discussing	8. Evaluation on test results

	your previously identified risks.	
7	**D2** Evaluate any new insights, ideas or potential improvements to your system and justify the reasons why you have chosen to include (or not to include) them as part of this business application.	9. A presentation about your application
8	**D3** Critically evaluate the strengths and weaknesses of your business application and fully justify opportunities for improvement and further development.	10. Evaluation of your application in the presentation

Submission Format
The submission is in the form of documents/files: 1. A report document including required evidences 2. An installable and executable version of your application (P5) 3. A presentation if necessary (D2, D3) You are required to make use of headings, paragraphs, subsections and illustrations as appropriate, and all work must be supported with research and referenced using the Harvard referencing system.

Table of content

1. Questionnaire for peer-review

1.1. Questions

1/ Would you think that the application could meet all the user requirements?

A. Yes

B. No

C. Other

2/ If these are requirement not meet, what requirement of which roles are not meet? (You could select multiple answers)

A. Admin

B. Training Staff

C. Trainer

D. Trainee

3/ What rate would you give for the solutions of the application for the requirement?

A. Very good

B. Good

C. Medium

D. Bad

E. Very bad

4/ What rate would you give for the interface of the application?

A. Very good

B. Good

C. Medium

D. Bad

E. Very bad

5/ Would you think the application need for improvement?

A. Yes

B. No

C. Other

6. What roles solution need to be improved? (You could select multiple answers)

A. Admin

B. Training Staff

C. Trainer

D. Trainee

1.2. Result

After conducting the questionnaire to peer-review, I have the following conclusion:

1/ The application meet the requirements of user requirement.

2/ The solutions and interfaces of the application rating is medium in average.

3/ The Training Staff solutions is considered need for improvement.

2. Application models

The application includes 10 models:

No	Models	Description
1	Account	The model contains attributes of login accounts.
2	User	The model contains basic attributes of all roles included in the system. The model is used for other role to inherit its attributes.
3	Staff	The model contain the attributes of Training Staff.
4	Trainer	The model contain the attributes of Trainer.
5	Trainee	The model contain the attributes of Trainee.
6	Subject	The model contains basic attributes of Category, Course, and Topic included in the system. The model is used for other class to inherit its attributes.
7	Category	The model contain the attributes of Category. It inherits attributes from Subject model.
8	Course	The model contain the attributes of Course. It inherits attributes from Subject model.
9	Topic	The model contain the attributes of Topic. It inherits attributes from Subject model.
10	CourseDetail	The model contain the foreign relationships with Course, Topic, and Trainer.

- **Account**

```
public class Account
{
    //Properties
    [Key]
    public int Id { get; set; }

    [MaxLength(20)]
    public string UserName { get; set; }


    [MaxLength(20)]
    public string PassWord { get; set; }

    [MaxLength(10)]
    public string Role { get; set; }

    public virtual Trainer Trainer { get; set; }
    public virtual Staff Staff { get; set; }
    public virtual Trainee Trainee { get; set; }
}
```

- **User**

```
public class User
{
    [ForeignKey("Account")]
    public int Id { get; set; }

    [MaxLength(20)]
    public string Name { get; set; }

    [MaxLength(6)]
    public string Gender { get; set; }

    public DateTime DoB { get; set; }

    [MaxLength(50)]
    public string Email { get; set; }

    [MaxLength(10)]
    public string Telephone { get; set; }

    [MaxLength(100)]
    public string Address { get; set; }


    public virtual Account Account { get; set; }
}
```

- **Staff**

```
public class Staff : User
{
}
```

- **Trainer**

```
public class Trainer : User
{
    [MaxLength(2)]
    public string Type { get; set; }

    [MaxLength(20)]
    public string Education { get; set; }

    [MaxLength(20)]
    public string Department { get; set; }

    [MaxLength(100)]
    public string WorkPlace { get; set; }

    public virtual ICollection<Topic> Topics { get; set; }
    public ICollection<CourseDetail> CourseDetails { get; set; }
}
```

- **Trainee**

```
public class Trainee : User
{
    [MaxLength(20)]
    public string Education { get; set; }

    [MaxLength(20)]
    public string MPL { get; set; }

    public float TOEICScore { get; set; }

    [MaxLength(100)]
    public string ExpDetail { get; set; }

    [MaxLength(20)]
    public string Department { get; set; }

    public virtual ICollection<Course> Courses { get; set; }
}
```

- **Subject**

```
public class Subject
{
    public int Id { get; set; }

    [MaxLength(20)]
    public string Name { get; set; }

    public string Description { get; set; }
}
```

- **Category**

```
public class Category : Subject
{
    public ICollection<Course> Courses { get; set; }

}
```

- **Course**

```
public class Course : Subject
{
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }

    public Category Category { get; set; }

    public virtual ICollection<Topic> Topics { get; set; }

    public virtual ICollection<Trainee> Trainees { get; set; }

    public ICollection<CourseDetail> CourseDetails { get; set; }
}
```

- **Topic**

```
public class Topic : Subject
{
    public virtual ICollection<Course> Courses { get; set; }
    public virtual ICollection<Trainer> Trainers { get; set; }
    public ICollection<CourseDetail> CourseDetails { get; set; }
}
```

- **CourseDetail**

```
public class CourseDetail
{

    public int Id { get; set; }

    public Course Course { get; set; }
    public Topic Topic { get; set; }
    public Trainer Trainer { get; set; }
}
```

3. Application controllers

The system includes 11 controllers:

No	Controllers	Description
1	AdminController	The controller take in charge all of functions when user login as Admin.
2	StaffController	The controller take in charge some basic functions when user login as Training Staff. The controller is used to inherit by other controller related to Training Staff.
3	StaffCategoryController	The controller take in charge all of functions about Category when user login as Training Staff.
4	StaffCourseController	The controller take in charge all of functions about Course when user login as Training Staff.

5	StaffTopicController	The controller take in charge all of functions about Topic when user login as Training Staff.
6	StaffTrainerController	The controller take in charge all of functions about Trainer when user login as Training Staff.
7	StaffTraineeController	TThe controller take in charge all of functions about Trainee when user login as Training Staff.
8	StaffProfileController	The controller take in charge all of functions about Training Staff profile when user login as Training Staff.
9	TrainerController	The controller take in charge all of functions when user login as Trainer.
10	TraineeController	The controller take in charge all of functions when user login as Trainee.
11	HomeController	The controller take in charge all of functions for login and log out of the application.

3.1. AdminController

```
 [Authorize]
public ActionResult DeleteAccount(int id, string role = "All", string search_name = "")
{
    CheckRole();

    Account account = db.Accounts.Find(id);
```

```
string r = account.Role;


switch (r)
{
    case "Admin":
        break;
    case "Staff":
        Staff staff = db.Staffs
                        .Where(s => s.Account.Id == id)
                        .FirstOrDefault();
        db.Staffs.Remove(staff);
        break;
    case "Trainer":
        Trainer trainer = db.Trainers
                        .Where(tr => tr.Account.Id == id)
                        .FirstOrDefault();
        db.Trainers.Remove(trainer);
        break;
    case "Trainee":
        Trainee trainee = db.Trainees
                        .Where(tr => tr.Account.Id == id)
                        .FirstOrDefault();
        db.Trainees.Remove(trainee);
        break;
    default:
        break;
}
db.Accounts.Remove(account);
```

```
            db.SaveChanges();

            return RedirectToAction("Index", "Admin", new
            {
                id = id,
                role = role,
                search_name = search_name
            });
        }
        [Authorize]
        public ActionResult AddNew(string username, string password, string repassword, string role)
        {
            CheckRole();

            Account account = db.Accounts.Where(a => a.UserName == username).FirstOrDefault();

            if(account != null)
            {
                TempData["result"] = "The username is already exist!";
                return RedirectToAction("AddNewInterface", "Admin");
            }
            account = new Account();
            account.UserName = username;
            account.PassWord = password;
            account.Role = role;

            db.Accounts.Add(account);
            switch (role)
            {
```

```
            case "Admin":
                break;

            case "Staff":
                Staff staff = new Staff();
                staff.Account = account;
                staff.Name = username;
                staff.Gender = "male";
                staff.DoB = Convert.ToDateTime("01/01/1990");
                db.Staffs.Add(staff);
                break;
            case "Trainer":
                Trainer trainer = new Trainer();
                trainer.Account = account;
                trainer.Name = username;
                trainer.Gender = "male";
                trainer.DoB = Convert.ToDateTime("01/01/1990");
                db.Trainers.Add(trainer);
                break;
            case "Trainee":
                Trainee trainee = new Trainee();
                trainee.Account = account;
                trainee.Name = username;
                trainee.Gender = "male";
                trainee.DoB = Convert.ToDateTime("01/01/1990");
                trainee.Department = "";
                trainee.MPL = "";
                trainee.TOEICScore = 0;
                db.Trainees.Add(trainee);
```

```
                break;
            default:

                break;
        }

        db.SaveChanges();
        return RedirectToAction("Index", "Admin");
    }
    [Authorize]
    public ActionResult Edit(int id, string username, string password, string repassword)
    {
        CheckRole();
        Account editAccount = db.Accounts.Find(id);

        if (editAccount == null)
        {
            TempData["result"] = "The modified user is not exist anymore!";
            return RedirectToAction("EditInterface", "Admin");
        }

        Account ifExistUsernameAccount = db.Accounts.Where(a => a.UserName == username).FirstOrDefault();
        if(ifExistUsernameAccount != null && editAccount.Id != ifExistUsernameAccount.Id)
        {
            TempData["result"] = "The username is taken!";
            return RedirectToAction("EditInterface", "Admin", new { id = id });
        }

        editAccount.UserName = username;
```

```
            editAccount.Password = password;
            TempData["result"] = "The data is modified!";

            db.SaveChanges();
            return RedirectToAction("EditInterface", "Admin", new { id = id });
        }
```

No	Method	Task
1	DeleteAccount	The method help admin to delete the existing account.
2	AddNew	The method help admin to add the new account into system
3	Edit	The method help admin to edit the existing account in the system

3.2. StaffController

```
public class StaffController : Controller
    {
        // GET: Staff
        protected AssignmentContext db = new AssignmentContext();

        [Authorize]
```

```
    protected ActionResult CheckRole()
    {
        try
        {
            if (Session["Role"].ToString() != "Staff")
                return RedirectToAction("Index", "Home");
            else
                return null;
        }
        catch
        {
            return RedirectToAction("Index", "Home");
        }
    }

    [Authorize]
    protected int ReturnId()
    {
        string id = Session["id"].ToString();
        int staffId = db.Database.SqlQuery<int>("SELECT Id FROM Staffs WHERE AccountId = " + id).FirstOrDefault();
        return staffId;
    }
}
```

No	Method	Task

1	CheckRole	Check the role of login user
2	ReturnId	Return the id of login training staff

3.3. StaffCategoryController

```
[Authorize]
        public ActionResult AddNew(string name, string description)
        {
            CheckRole();

            Category category = db.Categories.Where(a => a.Name == name).FirstOrDefault();
            if (category != null)
            {
                TempData["result"] = "The category name is already exist!";
                return RedirectToAction("Index", "StaffCategory");
            }

            category = new Category
            {
                Name = name,
                Description = description
            };

            db.Categories.Add(category);
```

```
            db.SaveChanges();
            TempData["result"] = "The category is added into the application!";

            return RedirectToAction("AddNewInterface", "StaffCategory");
        }

        [Authorize]
        public ActionResult Edit(int id, string name, string description)
        {
            CheckRole();
            Category EditCategory = db.Categories.Find(id);


            Category ifExistCategory = db.Categories.Where(a => a.Name == name).FirstOrDefault();
            if (ifExistCategory != null && EditCategory.Id != ifExistCategory.Id)
            {
                TempData["result"] = "The category name is taken!";
                return RedirectToAction("EditInterface", "StaffCategory", new { id = id });
            }

            EditCategory.Name = name;
            EditCategory.Description = description;
            TempData["result"] = "The category is modified!";
            db.SaveChanges();

            return RedirectToAction("EditInterface", "StaffCategory", new { id = id });
        }

[Authorize]
```

```
public ActionResult DeleteCategory(int id, string search_name)
{
    CheckRole();
    Category deleteCategory = db.Categories.Find(id);

    Category replaceCategory = db.Categories
                    .Where(c => c.Name == "No category")
                    .FirstOrDefault();

    if(replaceCategory == null)
    {
        replaceCategory = new Category();
        replaceCategory.Name = "No category";

        db.Categories.Add(replaceCategory);

        replaceCategory = db.Categories
                    .Where(c => c.Name == "No category")
                    .FirstOrDefault();
    }

    ICollection<Course> courses = db.Courses
                                .Where(c => c.Category.Id == id)
                                .ToList();

    foreach(Course course in courses)
    {
        course.Category = replaceCategory;
    }
```

```
        db.Categories.Remove(deleteCategory);

        db.SaveChanges();

        return RedirectToAction("Index", "StaffCategory", new { search_name = search_name});
    }

    [Authorize]

     public ActionResult RemoveCourse(int course_id, int category_id)
    {
        CheckRole();

        Course course = db.Courses.Find(course_id);

        Category replaceCategory = db.Categories
                          .Where(c => c.Name == "No category")
                          .FirstOrDefault();

        if (replaceCategory == null)
        {
            replaceCategory = new Category();
            replaceCategory.Name = "No category";

            db.Categories.Add(replaceCategory);

            replaceCategory = db.Categories
                          .Where(c => c.Name == "No category")
```

```
                        .FirstOrDefault();
        }

        course.Category = replaceCategory;
        db.SaveChanges();

        return RedirectToAction("EditInterface", "StaffCategory", new { id = category_id});
    }
```

No	Method	Task
1	DeleteCategory	The method help staff to delete the existing category.
2	AddNew	The method help staff to add the new category into system
3	Edit	The method help staff to edit the existing category in the system
4	RemoveCourse	The method help staff to remove the course from the editing category

3.4. StaffCourseController

```
[Authorize]
        public ActionResult DeleteCourse(int course_id, string search_name)
        {
            CheckRole();

            Course course = db.Courses.Find(course_id);

            ICollection<CourseDetail> details = db.CourseDetails
                                                .Where(d => d.Course.Id == course_id)
                                                .ToList();

            db.CourseDetails.RemoveRange(details);
            db.Courses.Remove(course);
            db.SaveChanges();

            return RedirectToAction("Index", "StaffCourse", new { search_name = search_name });
        }

        [Authorize]

          public ActionResult AddNew(string name, string category_name, string description, DateTime start_date, DateTime end_date)
        {
            CheckRole();

            Course course = db.Courses.Where(a => a.Name == name).FirstOrDefault();
            if (course != null)
```

```
        {
            TempData["result"] = "The course name is already exist!";
            return RedirectToAction("AddNewInterface", "StaffCourse");
        }

        Category category = db.Categories.Where(a => a.Name == category_name).FirstOrDefault();

        course = new Course();
        course.Name = name;
        course.Category = category;
        course.StartDate = start_date;
        course.EndDate = end_date;

        course.Description = description;

        db.Courses.Add(course);
        db.SaveChanges();
        int id = db.Courses.Max(c => c.Id);

        return RedirectToAction("EditInterface", "StaffCourse", new { id = id });
    }

    [Authorize]
    public ActionResult EditCourse(int id, string name, string category_name, string description)
    {
        CheckRole();

        Course editCourse = db.Courses.Find(id);
```

```
    Category selectedCategory = db.Categories
                                .Where(c => c.Name == category_name)
                                .FirstOrDefault();

    Course ifExistCourse = db.Courses.Where(a => a.Name == name).FirstOrDefault();
    if (ifExistCourse != null && editCourse.Id != ifExistCourse.Id)
    {
        TempData["result1"] = "The course name is taken!";
        return RedirectToAction("EditInterface", "StaffCourse", new { id = id });
    }

    editCourse.Name = name;
    editCourse.Description = description;
    editCourse.Category = selectedCategory;

    TempData["result1"] = "The course is modified!";
    db.SaveChanges();

    return RedirectToAction("EditInterface", "StaffCourse", new { id = id });
}


[Authorize]
public ActionResult AddTopic(int id, string topic_name, string trainer_name)
{
    CheckRole();

    CourseDetail detail = db.CourseDetails
                          .Where(d => d.Course.Id == id && d.Topic.Name == topic_name)
```

```
                        .FirstOrDefault();

if(detail != null)
 {
     TempData["result2"] = "The topic is already exist in this course!";
     return RedirectToAction("EditInterface", "StaffCourse", new { id = id });
 }
 else
 {
    Course course = db.Courses.Find(id);

    Topic topic = db.Topics
                  .Where(t => t.Name == topic_name)
                  .FirstOrDefault();

    Trainer trainer = db.Trainers
                      .Where(tr => tr.Name == trainer_name)
                      .FirstOrDefault();

    detail = new CourseDetail();
    detail.Course = course;
    detail.Topic = topic;
    detail.Trainer = trainer;

    db.CourseDetails.Add(detail);

    db.SaveChanges();

    TempData["result2"] = "The topic is added to this course successfully!";
```

```
            return RedirectToAction("EditInterface", "StaffCourse", new { id = id });
        }
    }

    [Authorize]
    public ActionResult RemoveTopic(int id, int detail_id)
    {
        CheckRole();

        CourseDetail detail = db.CourseDetails.Find(detail_id);
        db.CourseDetails.Remove(detail);
        db.SaveChanges();

        return RedirectToAction("EditInterface","StaffCourse", new {id  = id});
    }

    [Authorize]
    public ActionResult AddTrainee(int id, int trainee_id)
    {
        CheckRole();

        //Check if that trainee is already assigned to the course
        Trainee trainee = db.Courses
                            .Where(c => c.Id == id)
                            .SelectMany(c => c.Trainees)
                            .Where(tr => tr.Id == trainee_id)
                            .FirstOrDefault();

        if(trainee != null)
```

```
            {
                TempData["result3"] = "The trainee is already assigned to this course!";
                return RedirectToAction("EditInterface", "StaffCourse", new { id = id });

            }
            else
            {
                Course course = db.Courses.Find(id);
                trainee = db.Trainees.Find(trainee_id);

                if(trainee != null)
                {
                    course.Trainees.Add(trainee);
                    trainee.Courses.Add(course);

                    TempData["result3"] = "The trainee is assigned to this course successfully!";
                    db.SaveChanges();
                }
                else
                    TempData["result3"] = "This trainee is not exist!";


                return RedirectToAction("EditInterface", "StaffCourse", new { id = id });
            }

        }

        [Authorize]
        public ActionResult RemoveTrainee(int id, int trainee_id)
```

```
{
    CheckRole();

    Course course = db.Courses.Find(id);
    Trainee trainee = db.Trainees.Find(trainee_id);

    course.Trainees.Remove(trainee);
    trainee.Courses.Remove(course);

    db.SaveChanges();

    return RedirectToAction("EditInterface", "StaffCourse", new { id = id });
}
```

No	Method	Task
1	DeleteCourse	The method help staff to delete the existing course.
2	AddNew	The method help staff to add the new course into system

3	Edit	The method help staff to edit the existing course in the system
4	AddTopic	The method help staff to assign the topic into the editing course
5	RemoveTopic	The method help staff to remove the topic from the editing course
6	AddTrainee	The method help staff to assign the trainee into the editing course
7	RemoveTrainee	The method help staff to remove trainee from the editing course

3.5. StaffTopicController

```
[Authorize]
       public ActionResult DeleteTopic(int topic_id, string search_name)
       {
           CheckRole();

           Topic topic = db.Topics.Find(topic_id);

           ICollection<CourseDetail> details = db.CourseDetails
                                               .Where(d => d.Topic.Id == topic_id)
                                               .ToList();
```

```
            db.CourseDetails.RemoveRange(details);
            db.Topics.Remove(topic);
            db.SaveChanges();

            return RedirectToAction("Index", "StaffTopic", new { search_name  = search_name });
        }

        [Authorize]
        public ActionResult AddNew(string name, string description)
        {
            CheckRole();

            Topic topic = db.Topics.Where(a => a.Name == name).FirstOrDefault();
            if (topic != null)
            {
                TempData["result"] = "The topic name is already exist!";
                return RedirectToAction("AddNewInterface", "StaffTopic");
            }

            topic = new Topic
            {
                Name = name,
                Description = description
            };

            db.Topics.Add(topic);
            db.SaveChanges();

            int id = db.Topics.Max(c => c.Id);
```

```
        return RedirectToAction("EditInterface", "StaffTopic", new { id = id });
    }

    [Authorize]
    public ActionResult EditTopic(int id, string name, string description)
    {
        CheckRole();

        Topic EditTopic = db.Topics.Find(id);
        Topic ifExistTopic = db.Topics.Where(a => a.Name == name).FirstOrDefault();

        if (ifExistTopic != null && EditTopic.Id != ifExistTopic.Id)
        {
            TempData["result1"] = "The topic name is taken!";
            return RedirectToAction("EditInterface", "StaffTopic", new { id = id });
        }

        EditTopic.Name = name;
        EditTopic.Description = description;
        TempData["result1"] = "The topic is modified!";
        db.SaveChanges();
        return RedirectToAction("EditInterface", "StaffTopic", new { id = id });
    }

    [Authorize]
    public ActionResult EditTrainer(int topic_id, int trainer_id)
    {
        CheckRole();
        Trainer trainer = db.Topics
```

```
                        .Where(t => t.Id == topic_id)
                        .SelectMany(t => t.Trainers)
                        .Where(tr => tr.Id == trainer_id)
                        .FirstOrDefault();

        if(trainer != null)
        {
            TempData["result2"] = "The trainer is already assigned to this topic!";
            return RedirectToAction("EditInterface", "StaffTopic", new { id = topic_id });
        }

        trainer = db.Trainers.Find(trainer_id);

        if(trainer != null)
        {
            TempData["result2"] = "The trainer is assigned to this course successfully!";
            Topic topic = db.Topics.Find(topic_id);
            topic.Trainers.Add(trainer);
            trainer.Topics.Add(topic);
            db.SaveChanges();
        }
        else
            TempData["result2"] = "The trainer is not exist!";

        return RedirectToAction("EditInterface", "StaffTopic", new { id = topic_id });
    }

    [Authorize]
    public ActionResult RemoveTrainer(int topic_id, int trainer_id)
```

```
    {
        CheckRole();

        Topic topic = db.Topics.Find(topic_id);
        Trainer trainer = db.Trainers.Find(trainer_id);

        topic.Trainers.Remove(trainer);
        trainer.Topics.Remove(topic);

        db.SaveChanges();

        return RedirectToAction("EditInterface", "StaffTopic", new { id = topic_id});
    }
```

No	Method	Task
1	DeleteTopic	The method help staff to delete the existing topic.
2	AddNew	The method help staff to add the new topic into system
3	EditTopic	The method help staff to edit the existing topic in the system
4	EditTrainer	The method help staff to assign the trainer into the editing topic
5	RemoveTrainer	The method help staff to remove the trainer from the editing topic

3.6. StaffTrainerController

```
[Authorize]
       public ActionResult Edit(int id, string full_name, string gender, DateTime dob,
                                string email, string telephone, string address, string type,
                                string education, string work_place)
       {
           CheckRole();



           Trainer trainer = db.Trainers.Find(id);

           if (trainer != null)
           {
               trainer.Name = full_name;
               trainer.Gender = gender;
               trainer.DoB = dob;
               trainer.Email = email;
               trainer.Telephone = telephone;
               trainer.Address = address;
               trainer.Type = type;
               trainer.Education = education;
               trainer.WorkPlace = work_place;
           }

           db.SaveChanges();
```

```
            return RedirectToAction("EditInterface", "StaffTrainer", new { id = id });
        }
```

No	Method	Task
1	Edit	The method help staff to edit the existing trainer.

3.7. StaffTraineeController

```
[Authorize]
        public ActionResult DeleteTrainee(int trainee_id, int account_id, string full_name = "", string account_name = "", string mpl = "",
                                string department = "", int smaller_score = 0, int bigger_score = 1000)
        {
            CheckRole();

            Account account = db.Accounts.Find(account_id);
            Trainee trainee = db.Trainees.Find(trainee_id);

            db.Trainees.Remove(trainee);
            db.Accounts.Remove(account);
            db.SaveChanges();

            return RedirectToAction("Index", "StaffTrainee", new {
```

```
            full_name = full_name,
            account_name = account_name,
            mpl = mpl,
            department = department,
            smaller_score = smaller_score,
            bigger_score = bigger_score
        });
    }

    [Authorize]
    public ActionResult AddAccount(string username, string password)
    {
        CheckRole();

        Account account = db.Accounts.Where(a => a.UserName == username).FirstOrDefault();
        if (account != null)
        {
            TempData["result"] = "The username is already exist!";
            return RedirectToAction("AddAccountInterface", "StaffTrainee");
        }

        account = new Account();
        account.UserName = username;
        account.PassWord = password;
        account.Role = "Trainee";

        db.Accounts.Add(account);

        Trainee trainee = new Trainee();
```

```
        trainee.Account = account;
        trainee.Name = username;
        trainee.Gender = "male";
        trainee.DoB = Convert.ToDateTime("01/01/1990");
        trainee.Department = "";
        trainee.MPL = "";
        trainee.TOEICScore = 0;

        db.Trainees.Add(trainee);
        db.SaveChanges();

        int traineeId = db.Trainees.Max(c => c.Id);
        return RedirectToAction("EditProfileInterface", "StaffTrainee", new { id = traineeId });
    }



    [Authorize]
    public ActionResult EditAccount(int id, string username, string password)
    {
        CheckRole();
        Account editAccount = db.Accounts.Find(id);

        if (editAccount == null)
        {
            TempData["result"] = "The modified user is not exist anymore!";
            return RedirectToAction("EditInterface", "Admin");
        }
```

```
        Account ifExistUsernameAccount = db.Accounts.Where(a => a.UserName == username).FirstOrDefault();
        if (ifExistUsernameAccount != null && editAccount.Id != ifExistUsernameAccount.Id)
        {
            TempData["result"] = "The username is taken!";
            return RedirectToAction("EditAccountInterface", "StaffTrainee", new { id = id });
        }

        editAccount.UserName = username;
        editAccount.PassWord = password;
        TempData["result"] = "The account is modified!";
        db.SaveChanges();

        return RedirectToAction("EditAccountInterface", "StaffTrainee", new { id = id });
    }

    [Authorize]
    public ActionResult EditProfile(int id, string full_name, string gender, DateTime dob,
                            string email, string telephone, string mpl, string education,
                            int toeic_score, string department, string address)
    {
        CheckRole();
        Trainee trainee = db.Trainees.Find(id);

        if (trainee != null)
        {
            trainee.Name = full_name;
            trainee.Gender = gender;
            trainee.DoB = dob;
            trainee.Email = email;
```

```
            trainee.Telephone = telephone;
            trainee.MPL = mpl;
            trainee.Education = education;
            trainee.TOEICScore = toeic_score;
            trainee.Department = department;
            trainee.Address = address;
        }

        db.SaveChanges();
        return RedirectToAction("EditProfileInterface", "StaffTrainee", new { id = id });
    }
```

No	Method	Task
1	DeleteTrainee	The method help staff to delete the existing trainee.
2	AddAccount	The method help staff to add the new trainee account into system
3	EditAccount	The method help staff to edit the existing trainee account in the system
4	EditProfile	The method help staff to edit the existing trainee profile in the system

3.8. StaffProfileController

```
[Authorize]
      public ActionResult Edit(string full_name, string gender, DateTime dob, string email, string telephone, string address)
      {
          CheckRole();
          int staffId = ReturnId();

          Staff staff = db.Staffs.Find(staffId);

          if (staff != null)
          {
              staff.Name = full_name;
              staff.Gender = gender;
              staff.DoB = dob;
              staff.Email = email;
              staff.Telephone = telephone;
              staff.Address = address;
          }

          db.SaveChanges();
          return RedirectToAction("Index", "StaffProfile");
      }
```

No	Method	Task

1	Edit	The method help staff to edit the their own profile.

3.9. TrainerController

```
[Authorize]
        public ActionResult Profile()
        {
            CheckRole();
            int trainerId = ReturnId();

            Trainer trainer = db.Trainers.Find(trainerId);

            if(trainer == null)
            {
                trainer = new Trainer
                {
                    Id = 0,
                    Name = "",
                    Gender = "male",
                    DoB = System.DateTime.Now,
                    Email = "",
                    Telephone = "",
                    Address = "",
                    Type = "in",
                    Education = "",
                    WorkPlace = ""
```

```
            };
        }
        return View(trainer);
    }

    [Authorize]
    public ActionResult Edit(string full_name, string gender, DateTime dob, string email, string telephone, string address, string type, string education, string work_place)
    {
        CheckRole();
        int trainerId = ReturnId();

        Trainer trainer = db.Trainers.Find(trainerId);

        if(trainer != null)
        {

            trainer.Name = full_name;
            trainer.Gender = gender;
            trainer.DoB = dob;
            trainer.Email = email;
            trainer.Telephone = telephone;
            trainer.Address = address;
            trainer.Type = type;
            trainer.Education = education;
            trainer.WorkPlace = work_place;
        }

        db.SaveChanges();
```

```
        return RedirectToAction("Profile", "Trainer");
    }

    //Course detail
    [Authorize]
    public ActionResult CourseDetail(string name)
    {
        CheckRole();

        Course course = db.Courses
            .Where(c => c.Name == name)
            .Include(c => c.Category)
            .FirstOrDefault();

        var trainees = db.Courses.Where(c => c.Name == name)
                         .SelectMany(c => c.Trainees)
                         .Select(tr => new { Id = tr.Id, Name = tr.Name }).ToList();

        ViewBag.course = course;
        ViewBag.trainees = trainees;

        return View();
    }
```

No	Method	Task
1	Profile	The method display the profile of login trainer.

2	Edit	The method help login trainer to edit their own profile.
3	CourseDetail	The method display the detail of course of login trainer is assigned to.

3.10. TraineeController

```
[Authorize]
        public ActionResult Profile()
        {
            CheckRole();
            int traineeId = ReturnId();

            Trainee trainee = db.Trainees.Find(traineeId);

            return View(trainee);
        }
        [Authorize]
        public ActionResult CourseDetail(int id)
        {
            CheckRole();
            Course course = db.Courses.Where(c => c.Id == id).Include(c => c.Category).FirstOrDefault();
            var topics = db.CourseDetails
                .Where(d => d.Course.Id == id)
                .Include(d => d.Topic)
                .Include(d => d.Trainer)
```

```
            .Select(d => new
            {
                TopicName = d.Topic.Name,
                TrainerName = d.Trainer.Name,
                Description = d.Topic.Description
            }).ToList();

        ViewBag.course = course;
        ViewBag.topics = topics;
        return View();
    }
```

No	Method	Task
	Profile	The method display the profile of login trainee.
	CourseDetail	The method display the detail of course of login trainee is assigned to.

3.11. HomeController

```
    [HttpPost]
    public ActionResult Login(string username, string password)
    {
        Account account = db.Accounts.Where(a => a.UserName == username && a.PassWord == password).FirstOrDefault();
```

```
        if(account != null)
        {
            Session["id"] = account.Id;
            Session["username"] = account.UserName;
            Session["Role"] = account.Role;

            FormsAuthentication.SetAuthCookie(account.UserName, false);
            switch (account.Role)
            {
                case "Admin":
                    return RedirectToAction("Index", "Admin");
                case "Staff":
                    return RedirectToAction("Index", "StaffProfile");
                case "Trainer":
                    return RedirectToAction("Index", "Trainer");
                case "Trainee":
                    return RedirectToAction("Index", "Trainee");
                default:
                    return RedirectToAction("About", "Home");
            }
        }
        else
        {
            TempData["result"] = "Username or password is invalid!";
            return RedirectToAction("Index", "Home");
        }

    }
```

```
[Authorize]
public ActionResult Logout()
{
    FormsAuthentication.SignOut();
    Session.Abandon();
    return RedirectToAction("Index", "Home");
}
```

No	Method	Task
1	Login	The method help user to login to the system.
2	Logout	The method help user to log out from the system.

4. Application interfaces

4.1. Login page

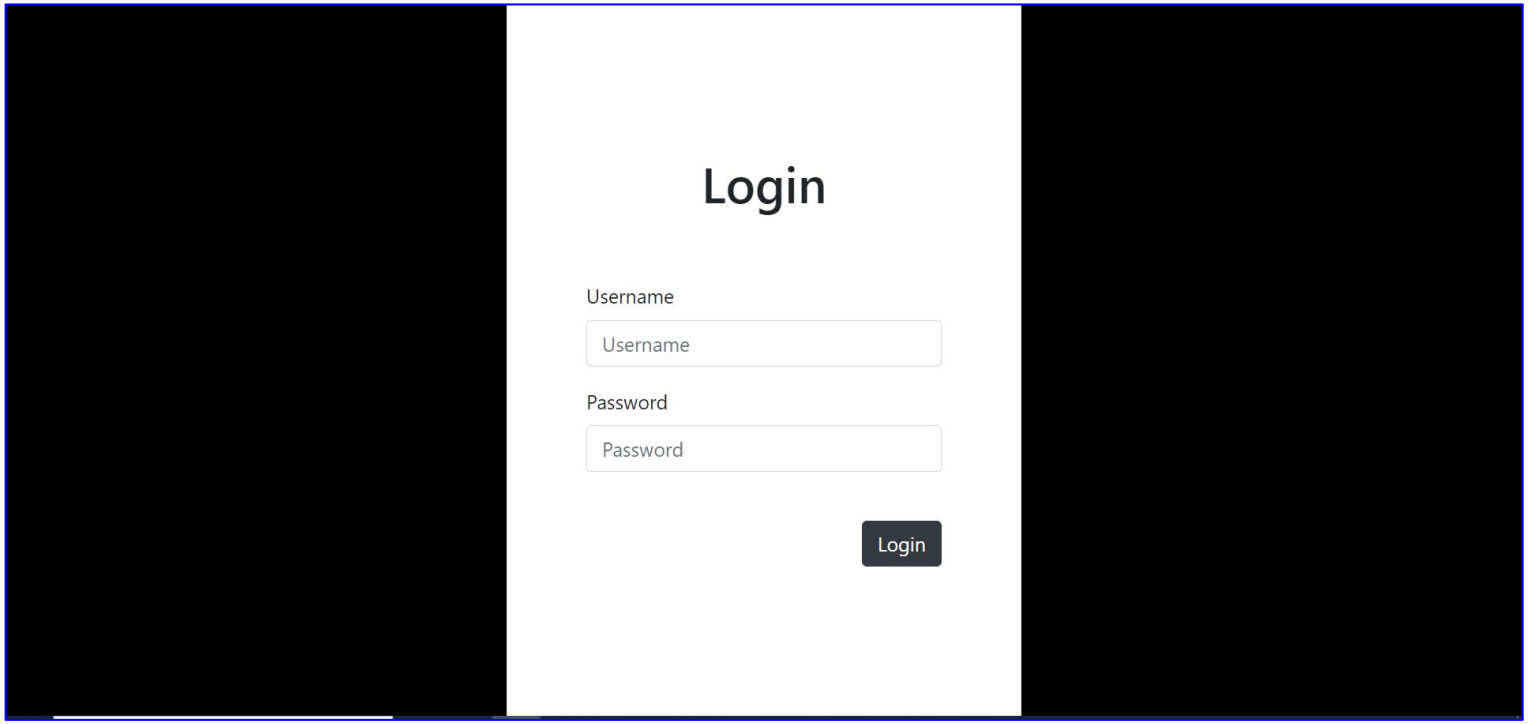

Figure 1 : Login page

4.2. Login as Admin

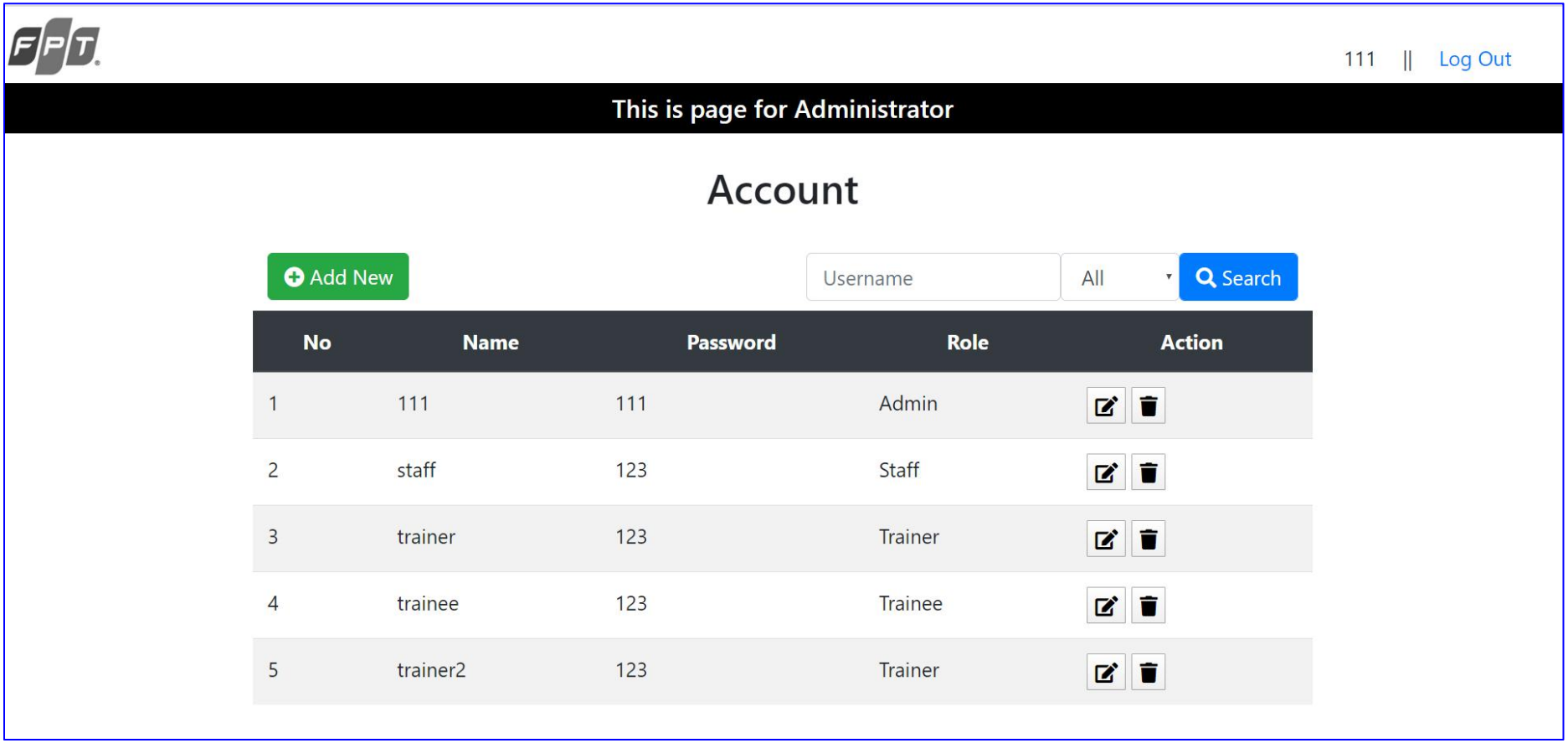

Figure 2 : The page show list of existing account

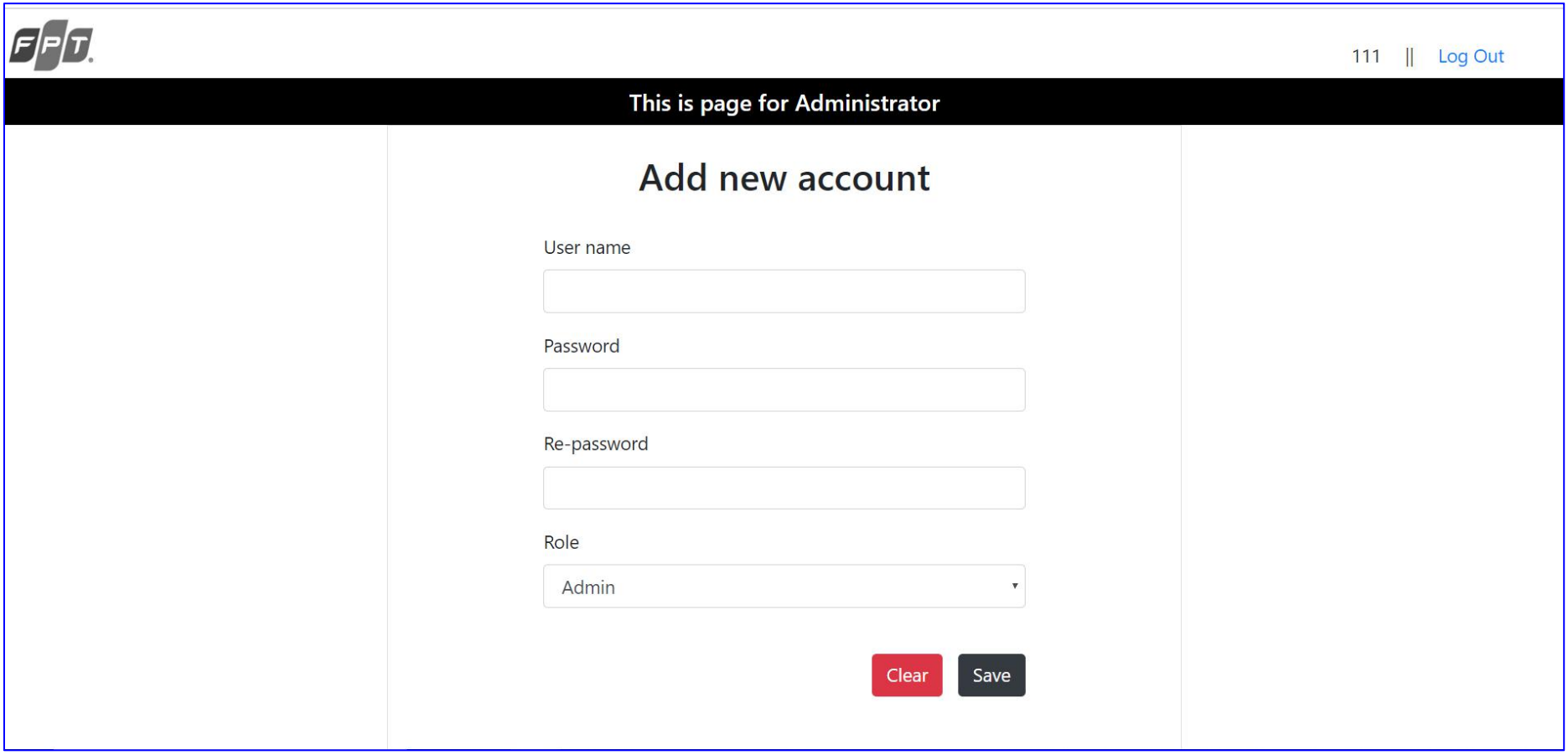

Figure 3 : The page for admin to add newt account

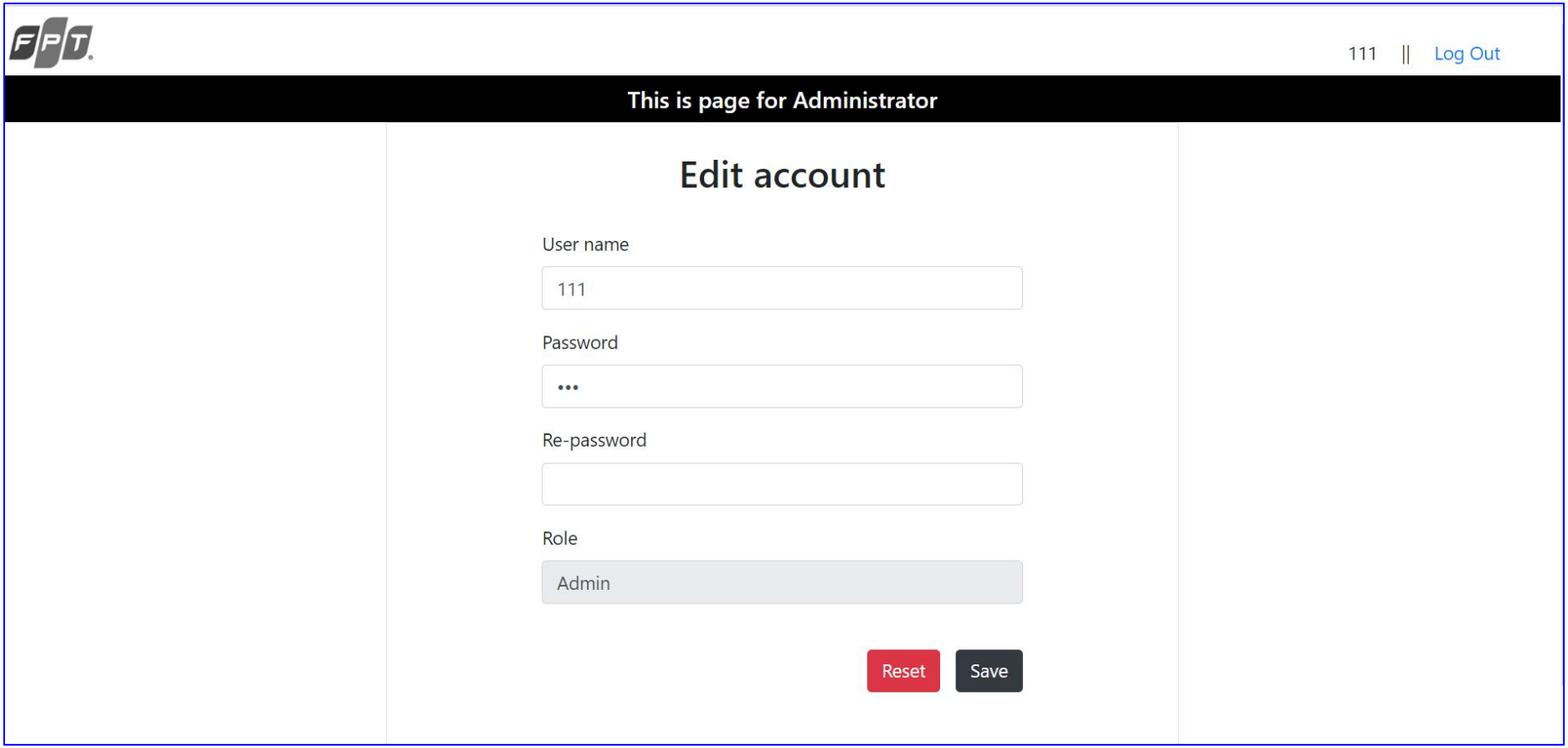

Figure 4 : The page for admin to edit account

4.3. Login as Training Staff

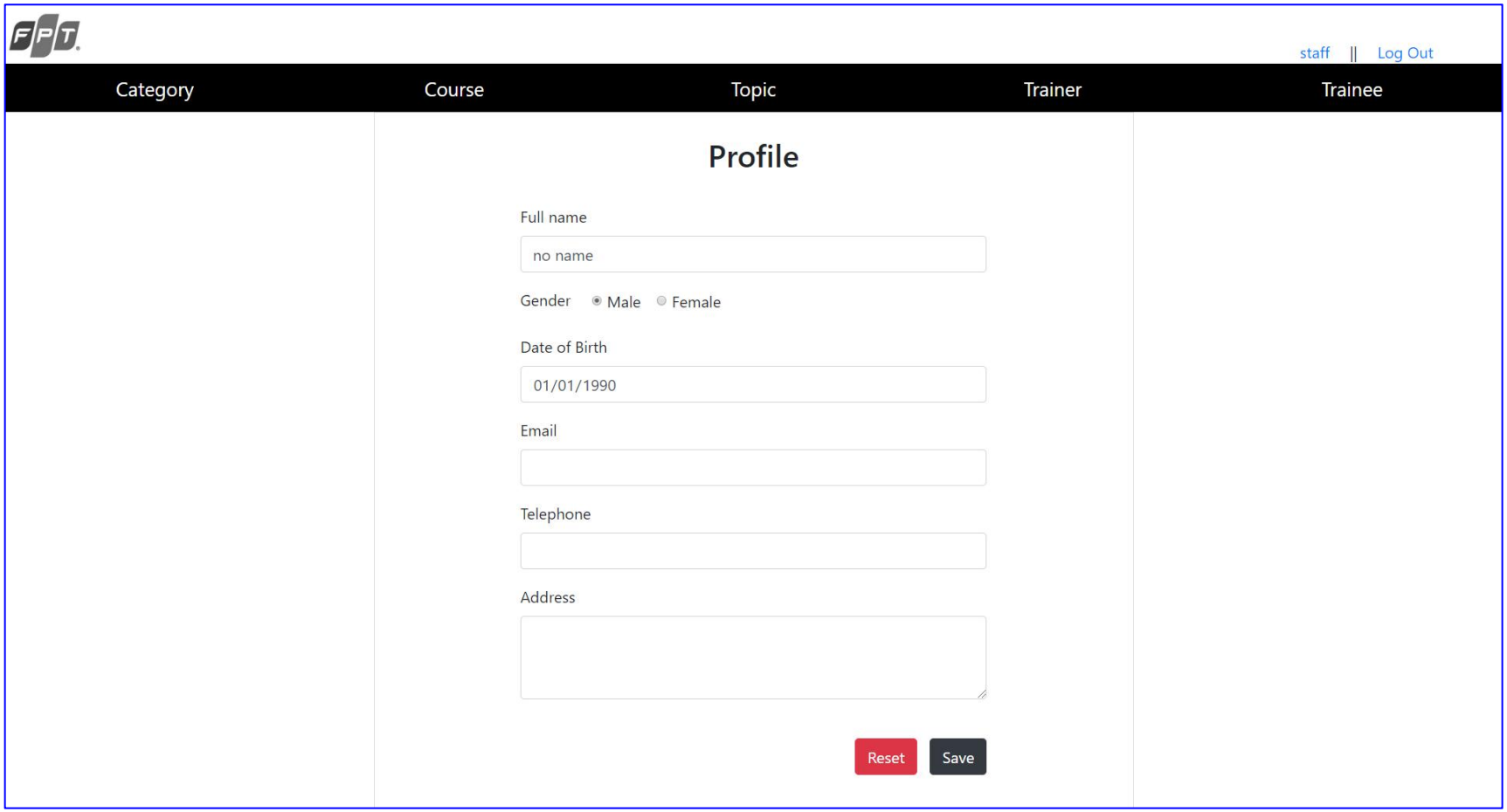

Figure 5 : The page for staff to edit their profile

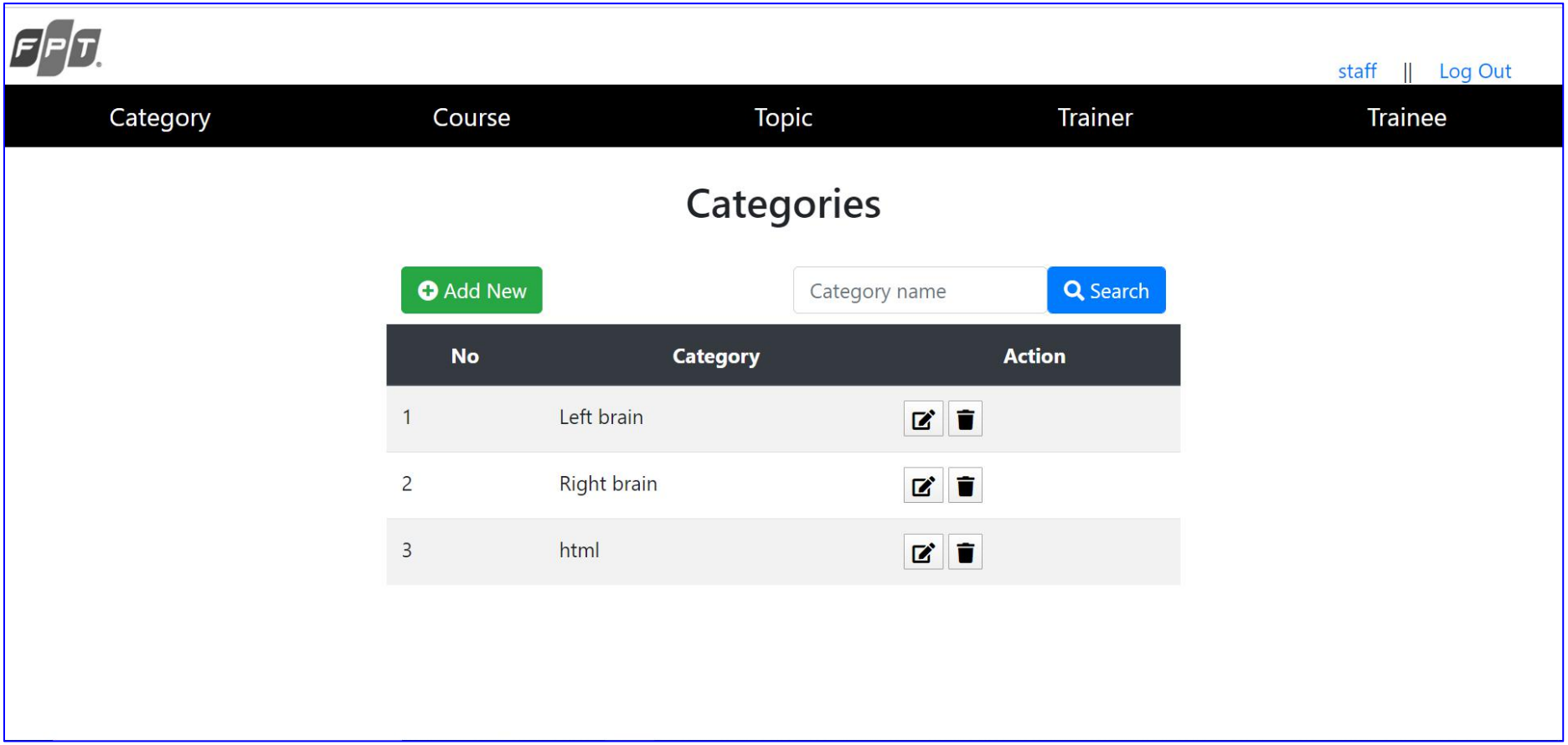

Figure 6 : The page show list of existing category

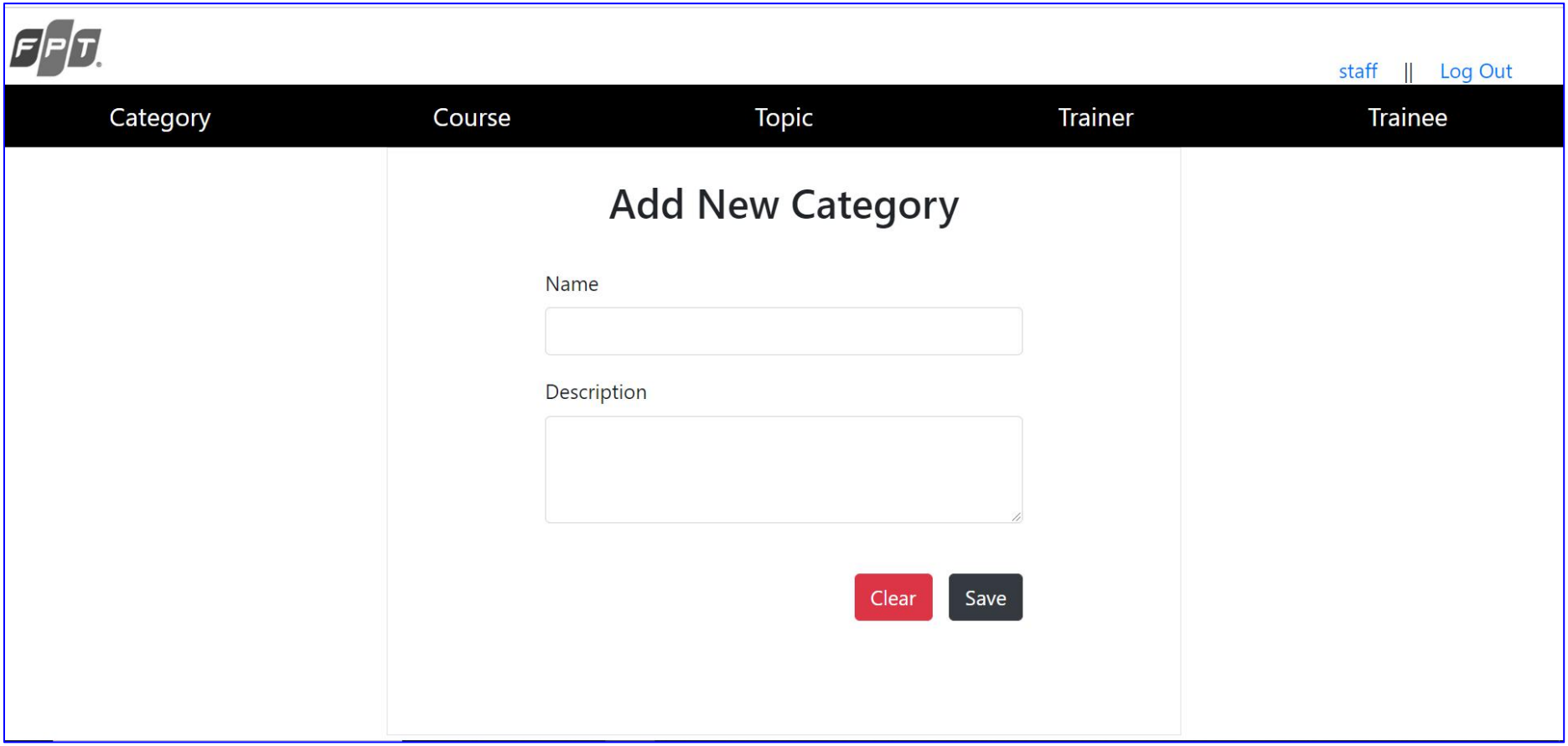

Figure 7 : The page for staff to add new category

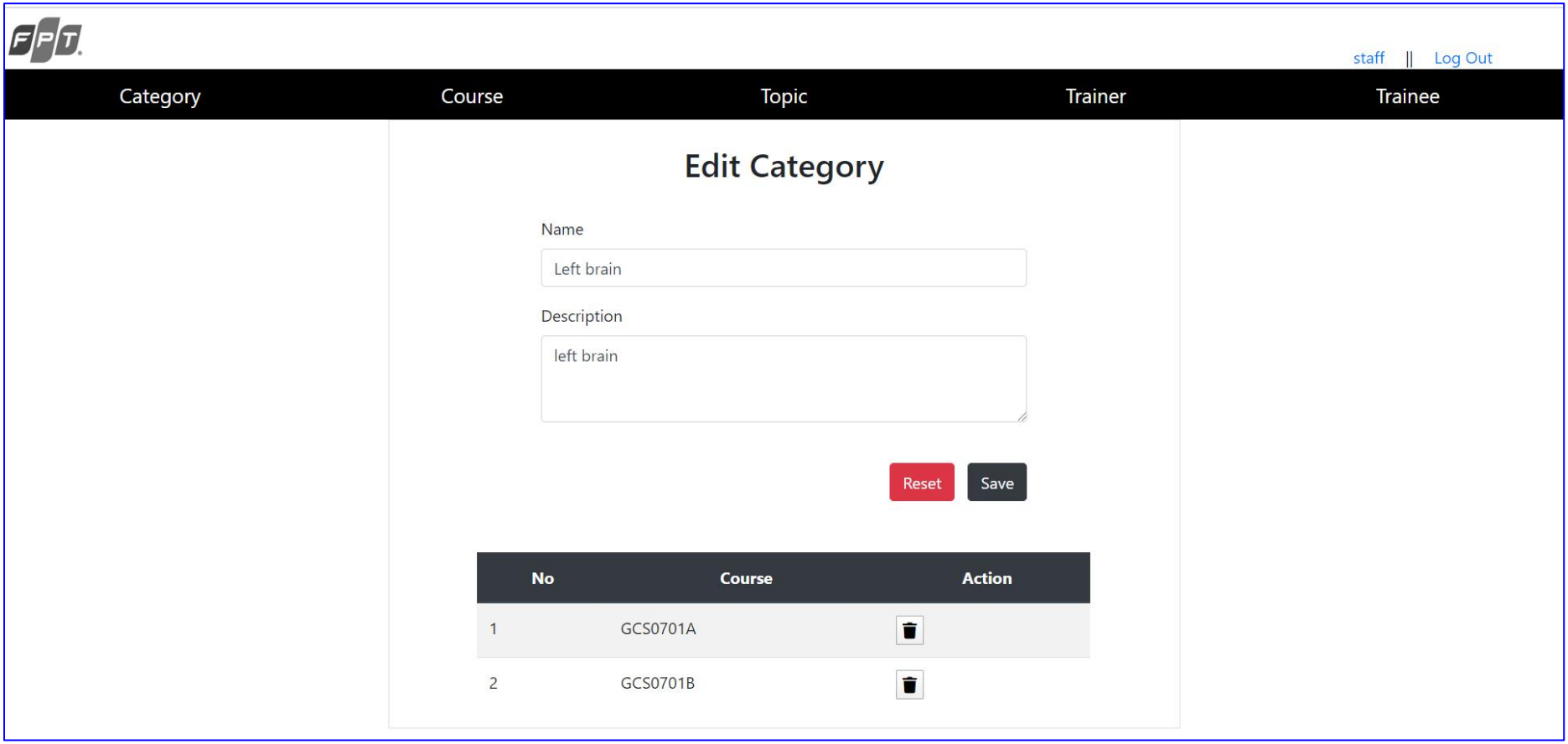

Figure 8 : The page for staff to edit category

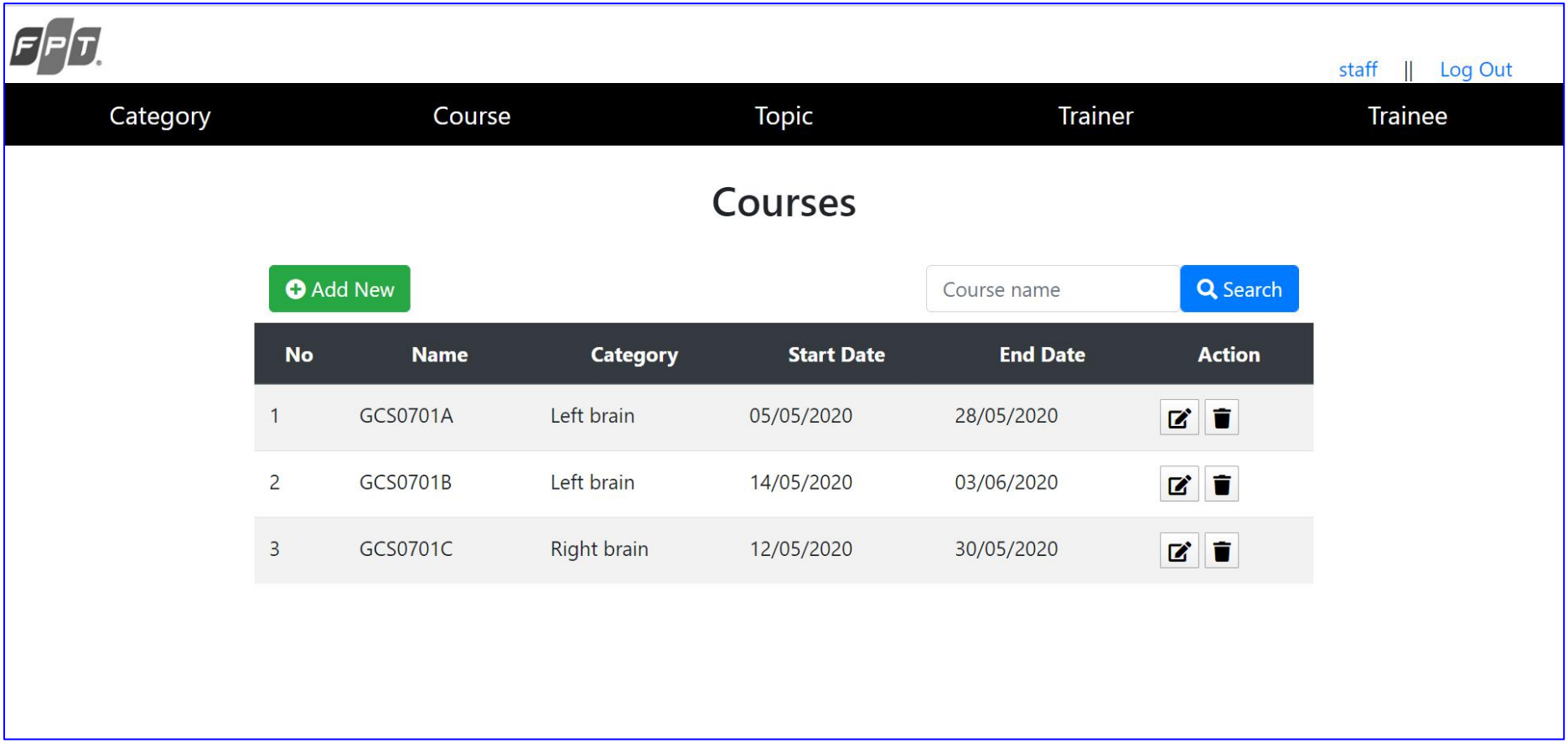

Figure 9 : The page show list of existing course

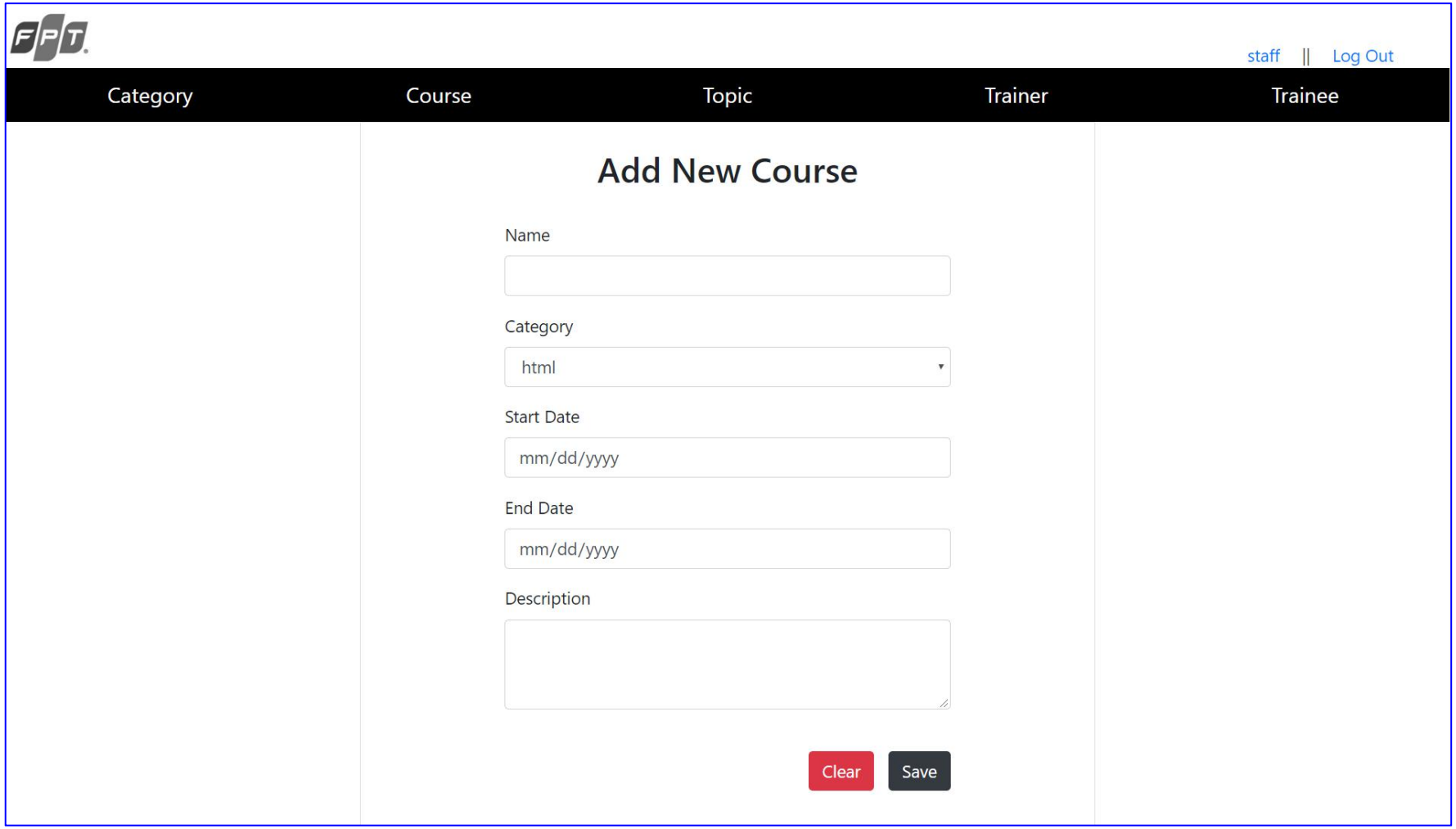

Figure 10 : The page for staff to add new course

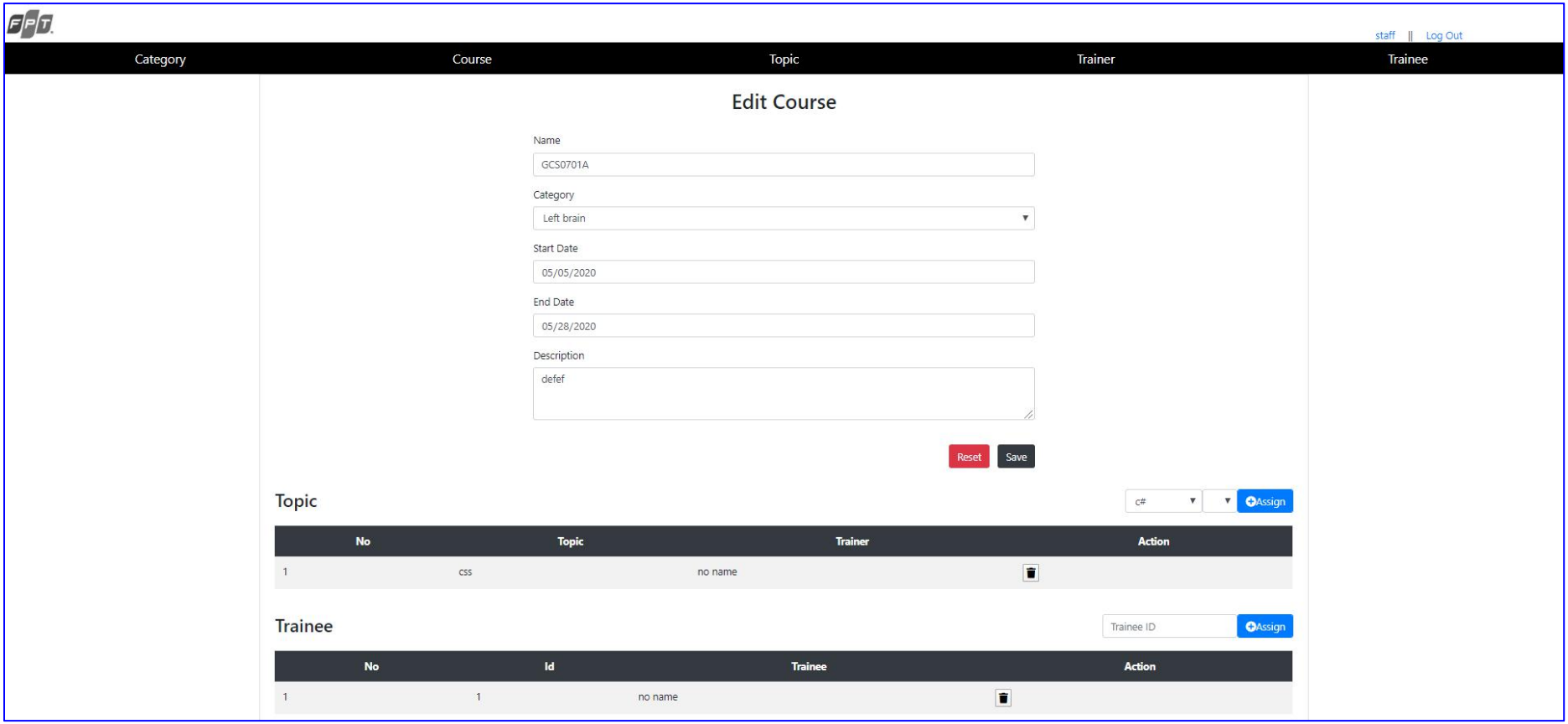

Figure 11 : The page for staff to edit course

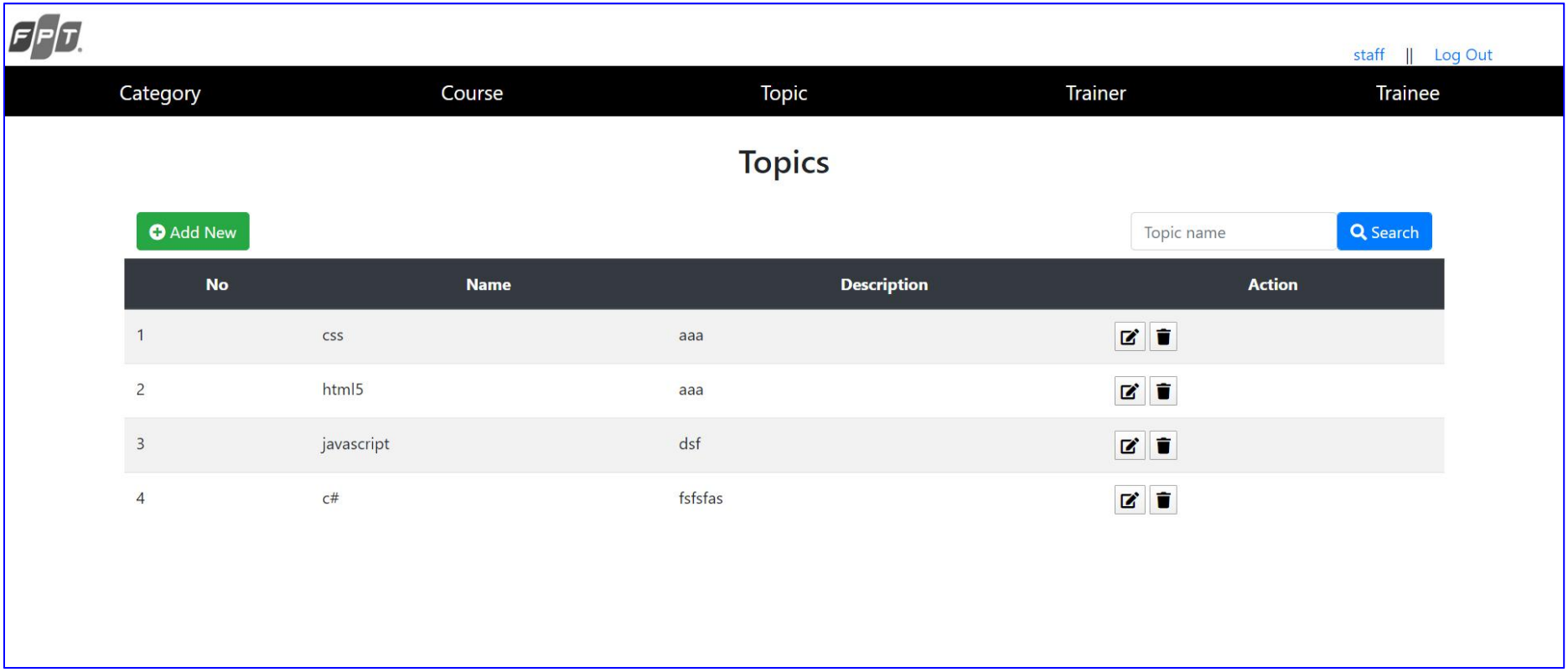

Figure 12 : The page show list of existing topic

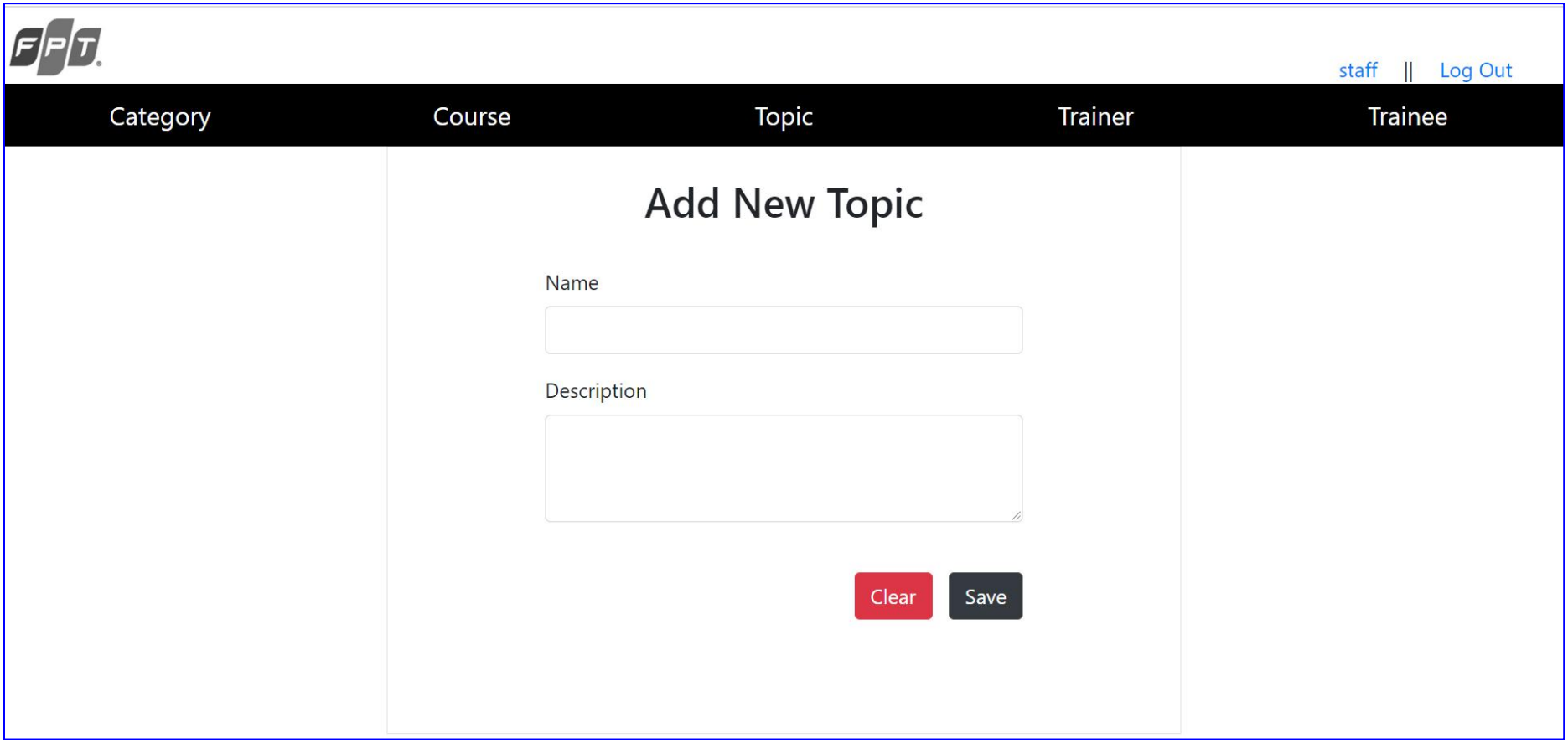

Figure 13 : The page for staff to add new topic

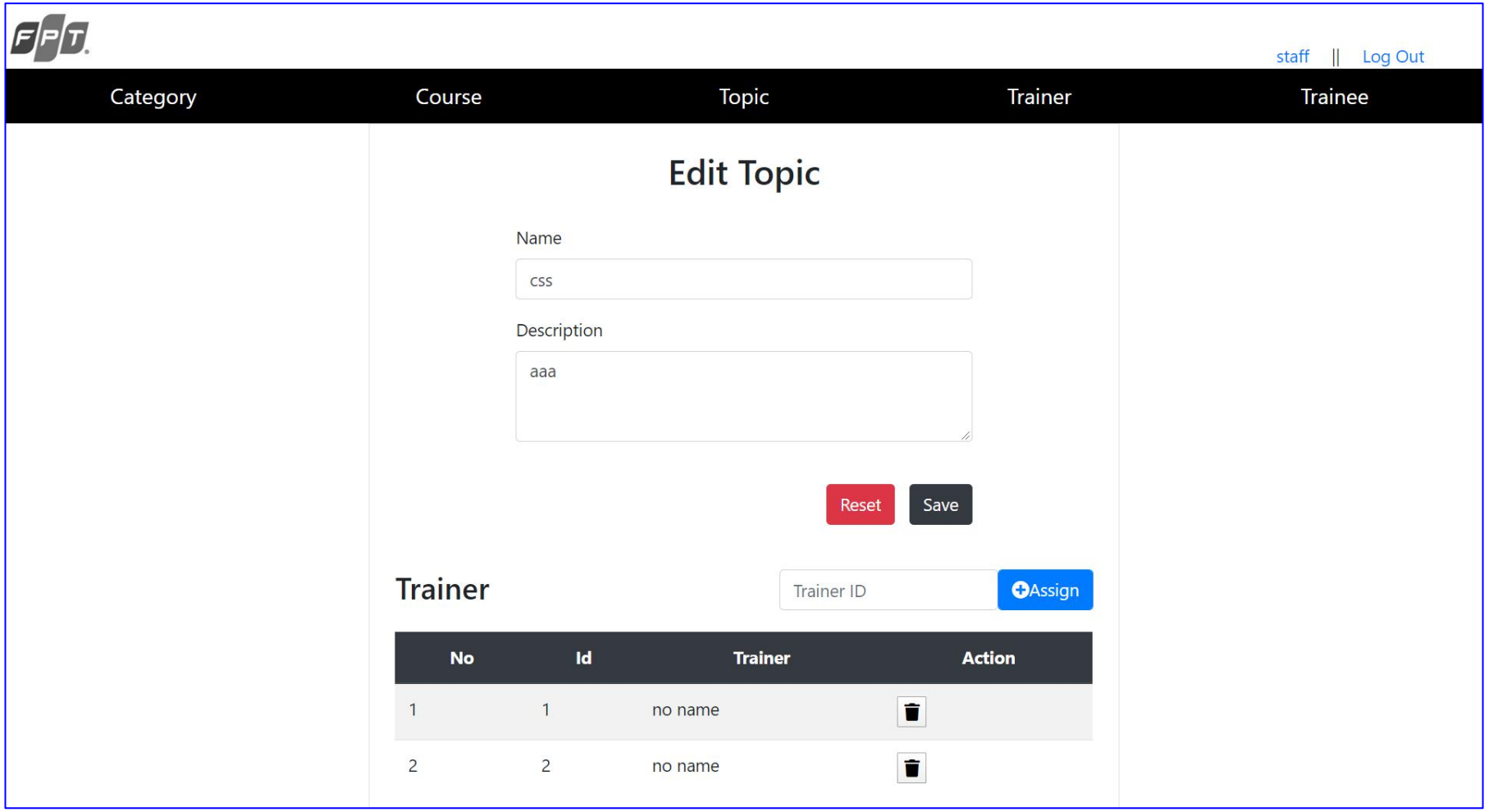

Figure 14 : The page for staff to edit topic

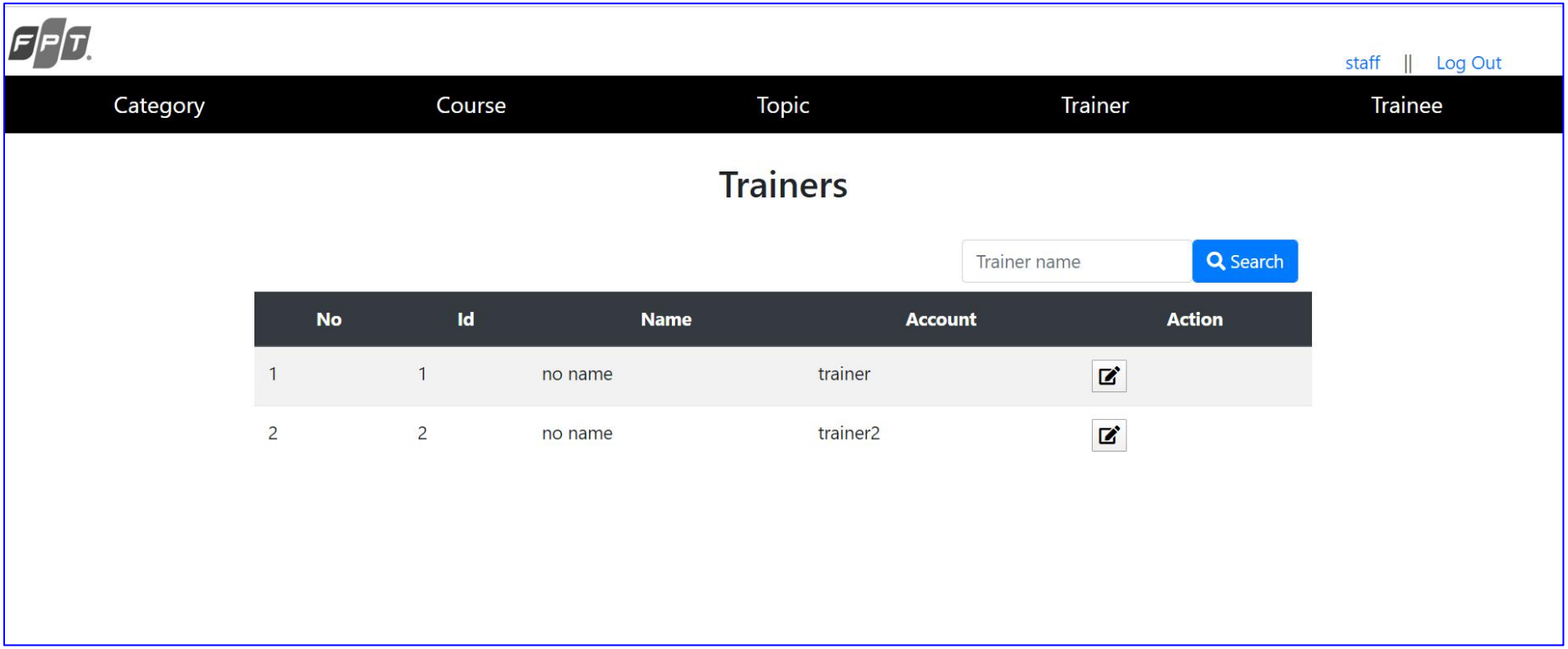

Figure 15 : The page show list of existing trainer

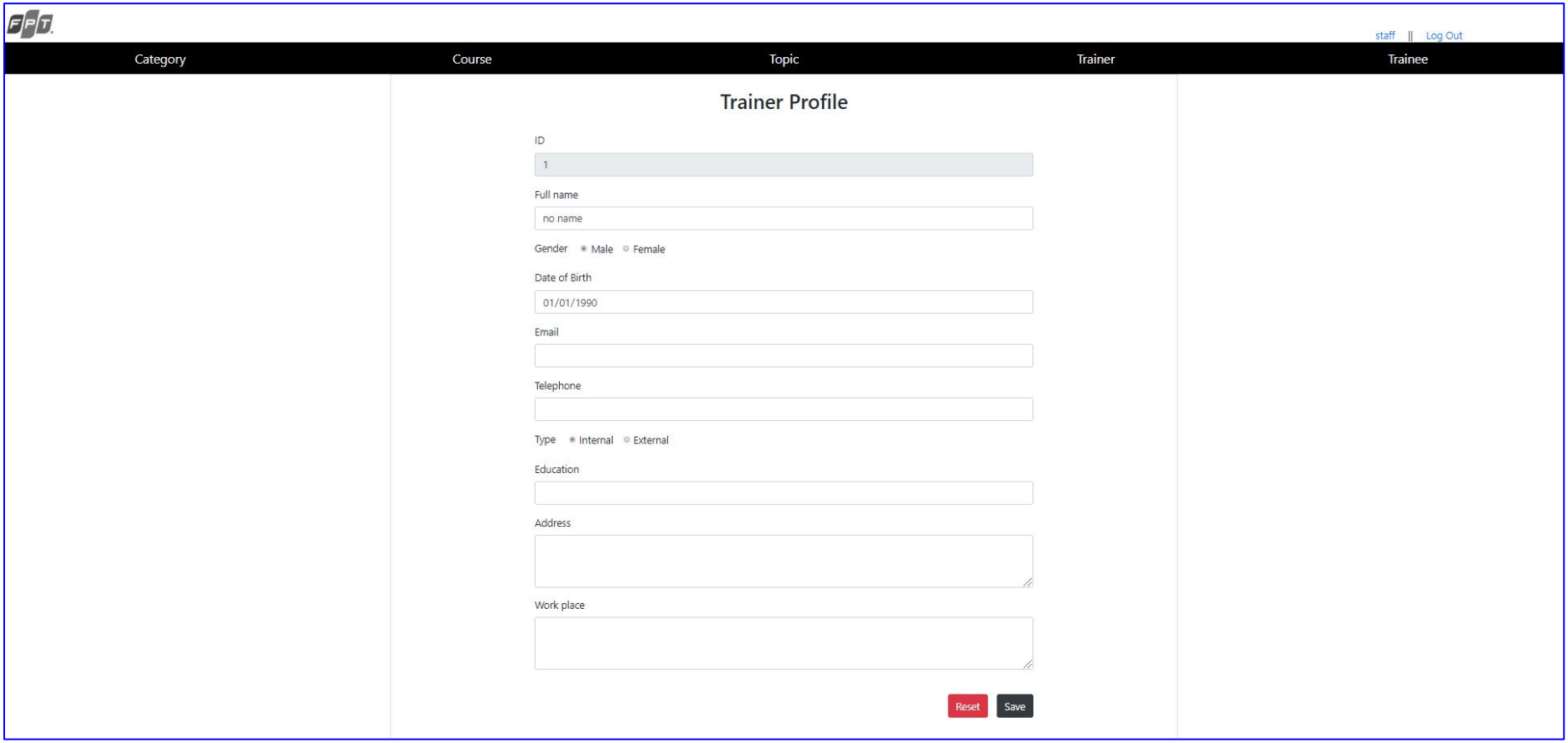

Figure 16 : The page for staff to edit trainer profile

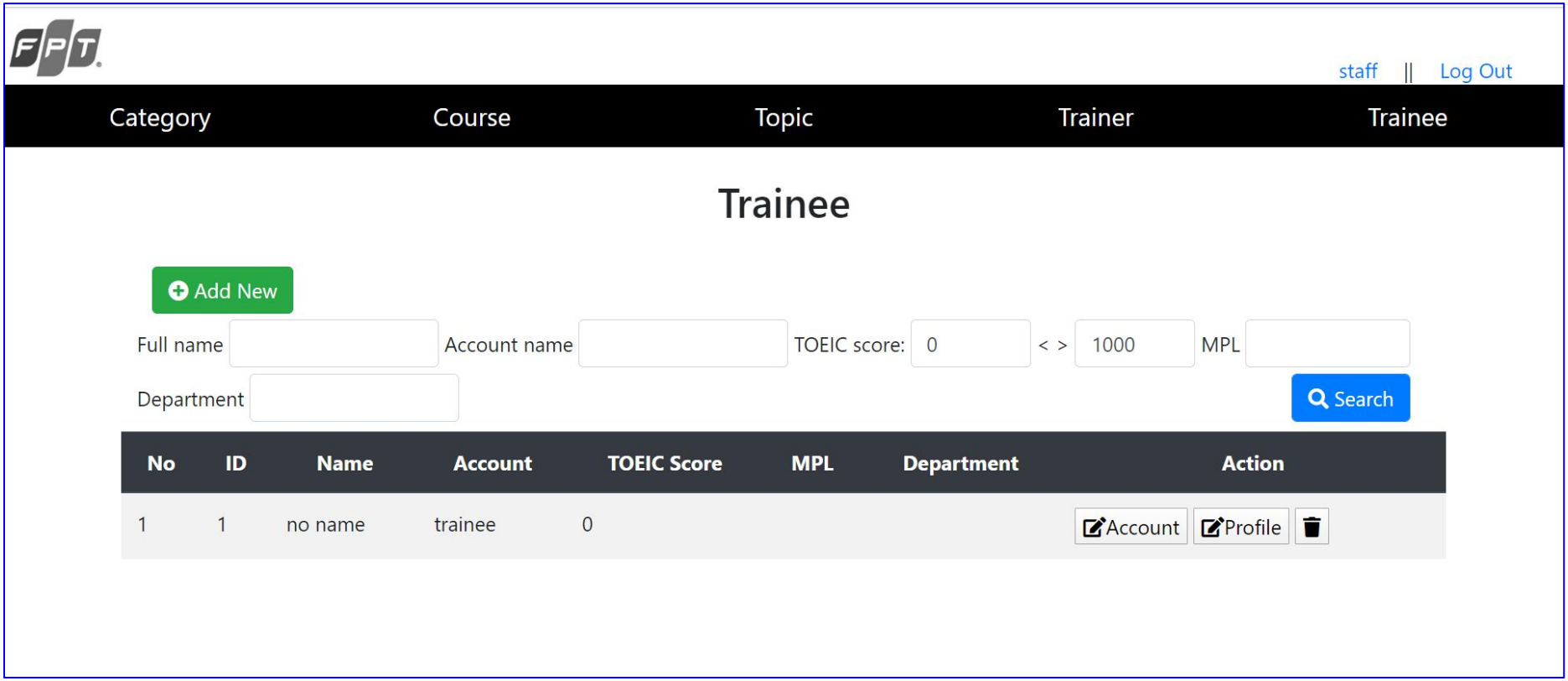

Figure 17: The page show list of existing trainee

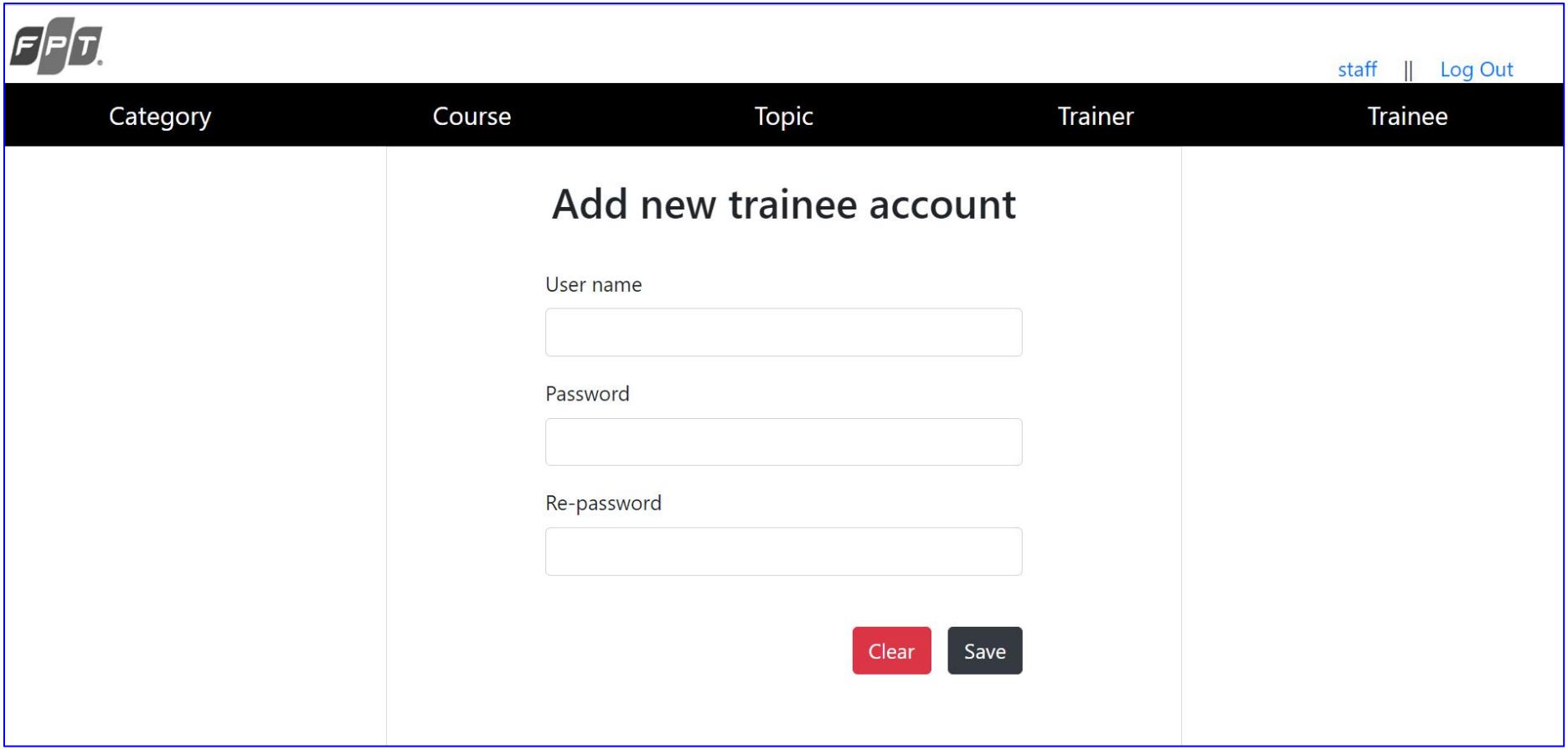

Figure 18 : The page for staff to add new trainee account

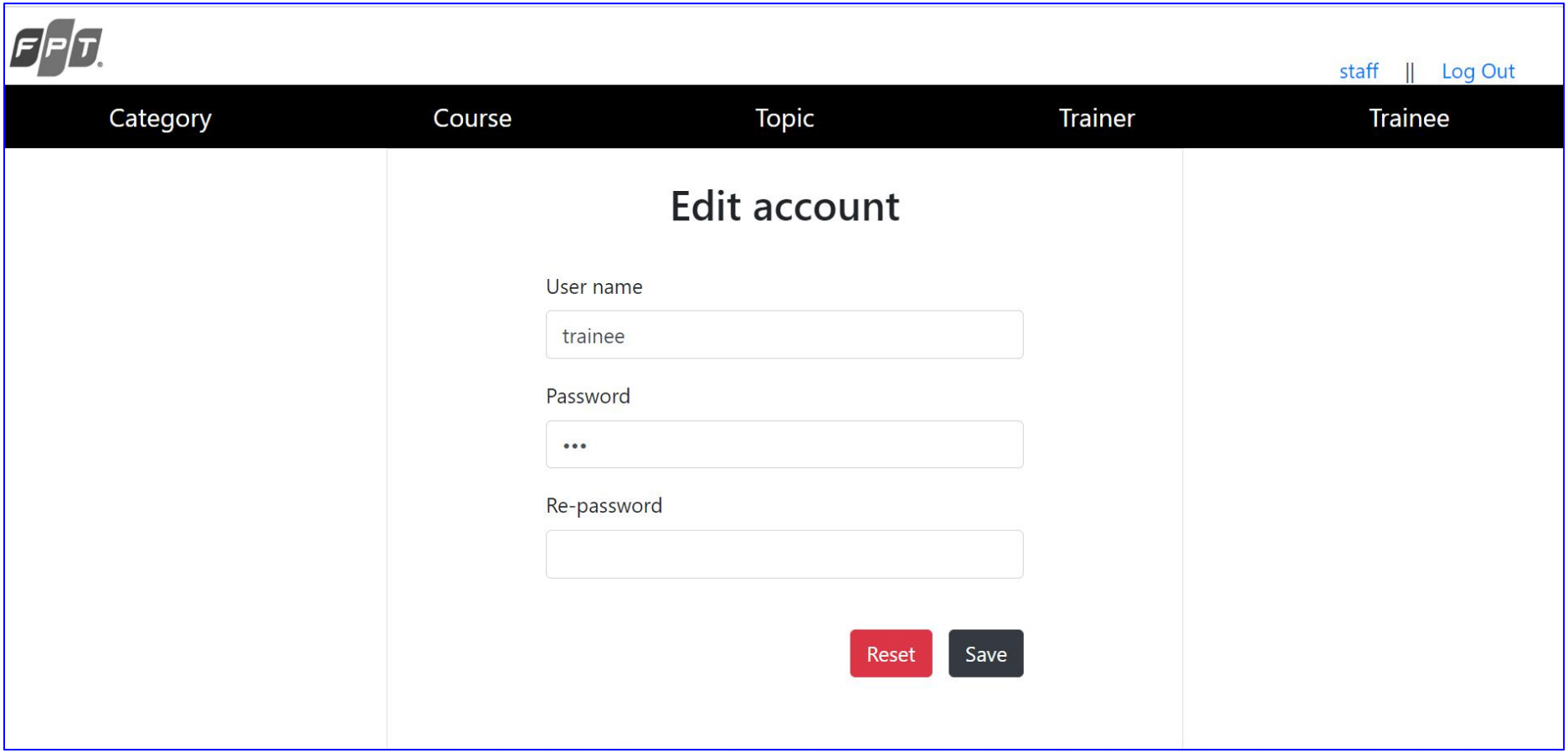

Figure 19 : The page for staff to edit trainee account

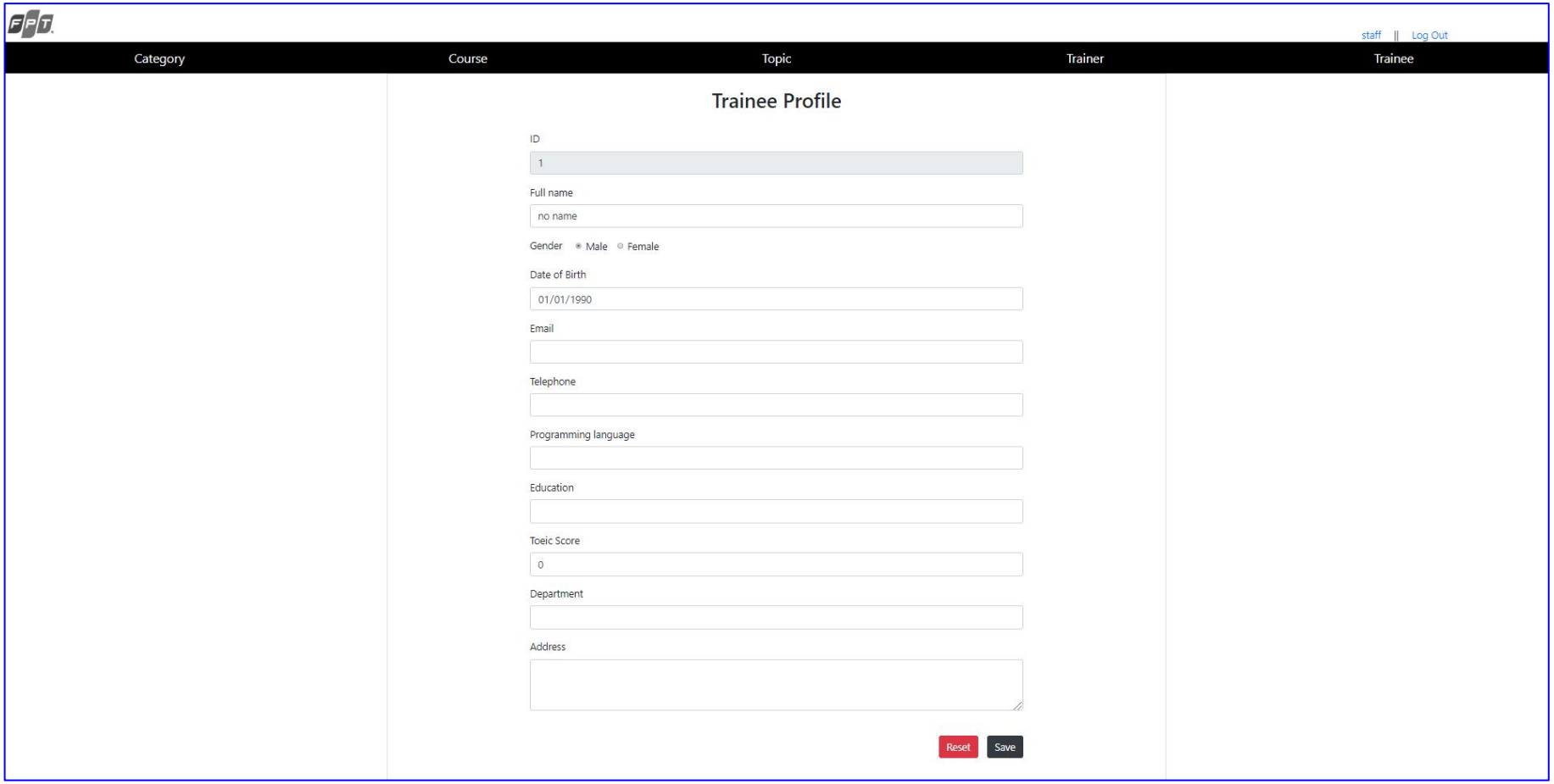

Figure 20 : The page for staff to edit trainee profile

4.4. Login as Trainer

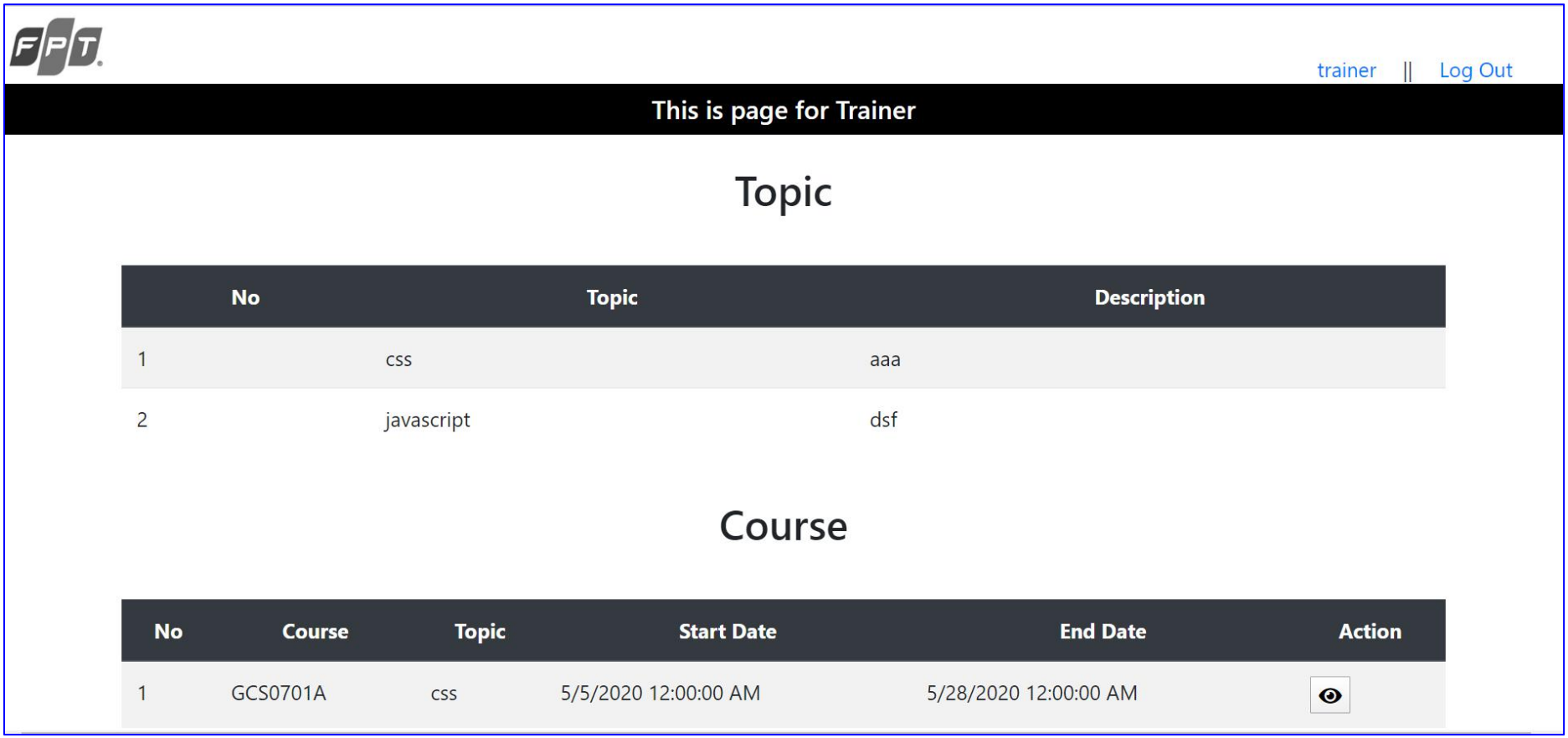

Figure 21 : The page show of assigned topics and courses of login trainer

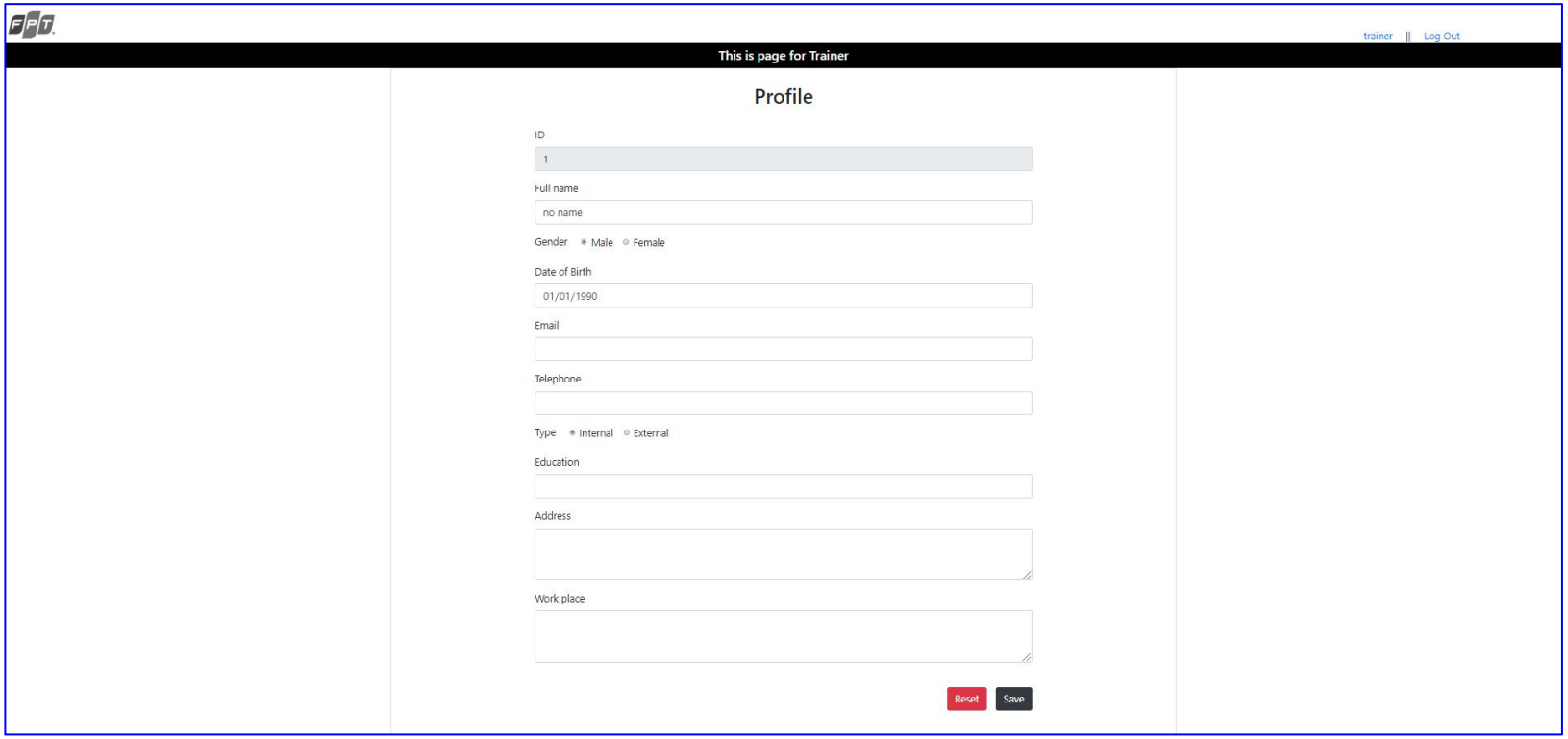

Figure 22 : The page for login trainer to edit their profile

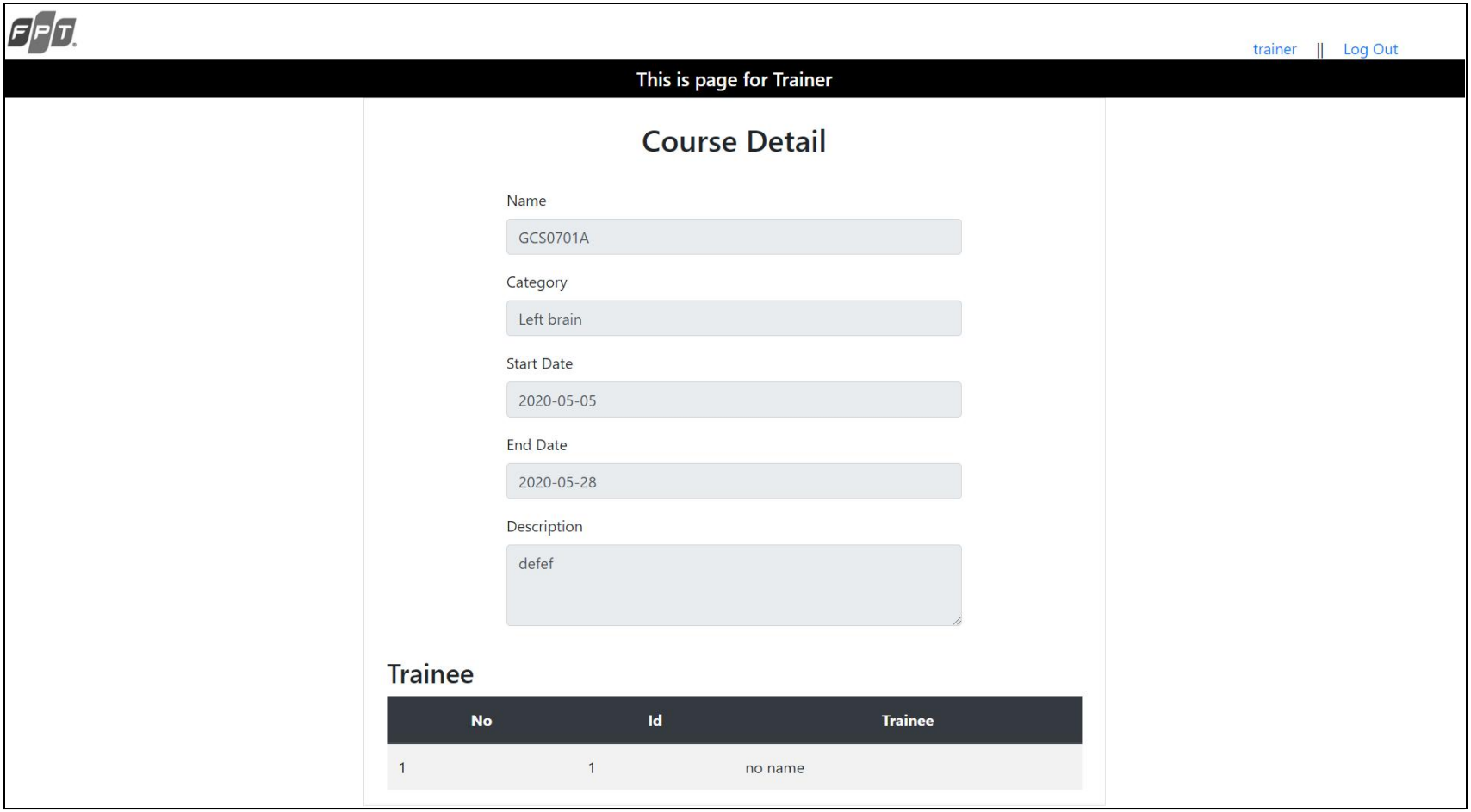

Figure 23 : The page show the detail of assigned course of login trainer

4.5. Login as Trainee

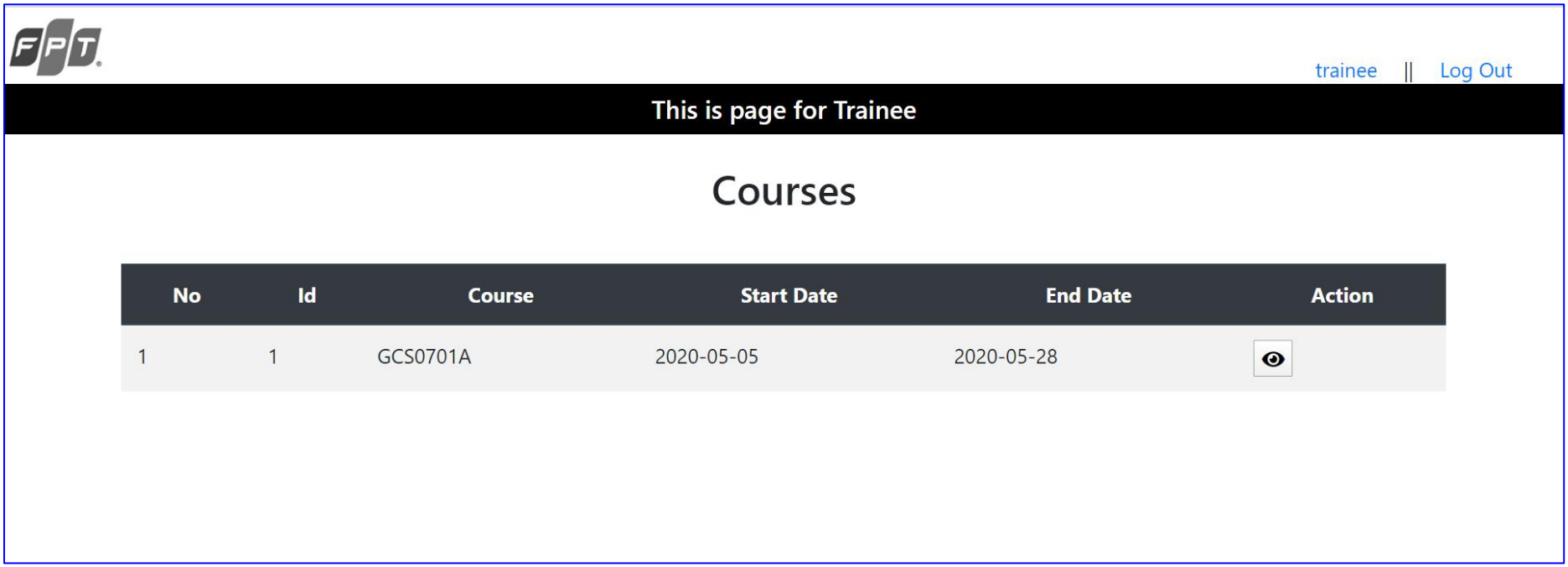

Figure 24 : The page show of assigned courses of login trainee

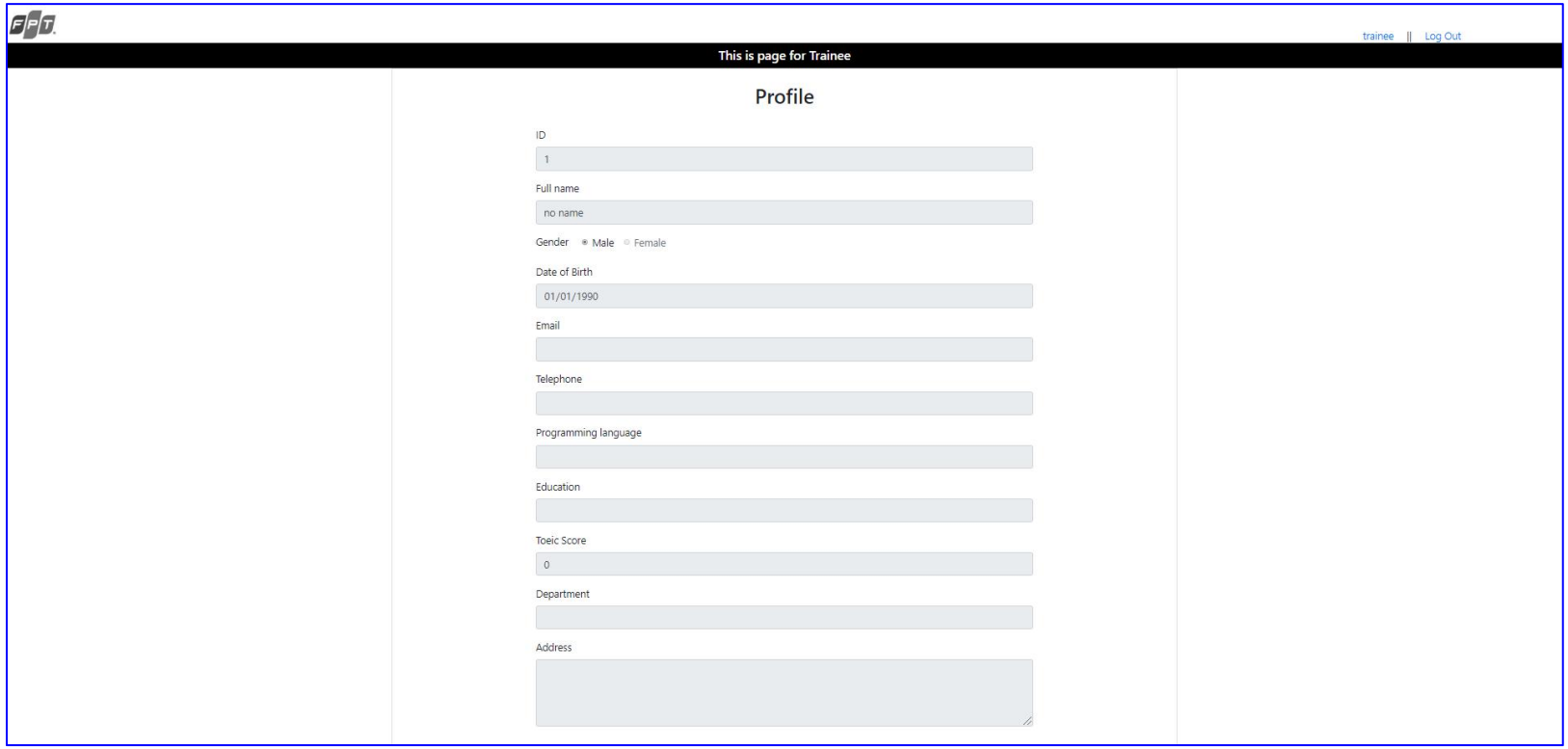

Figure 25 : The page show the profile of the login trainee

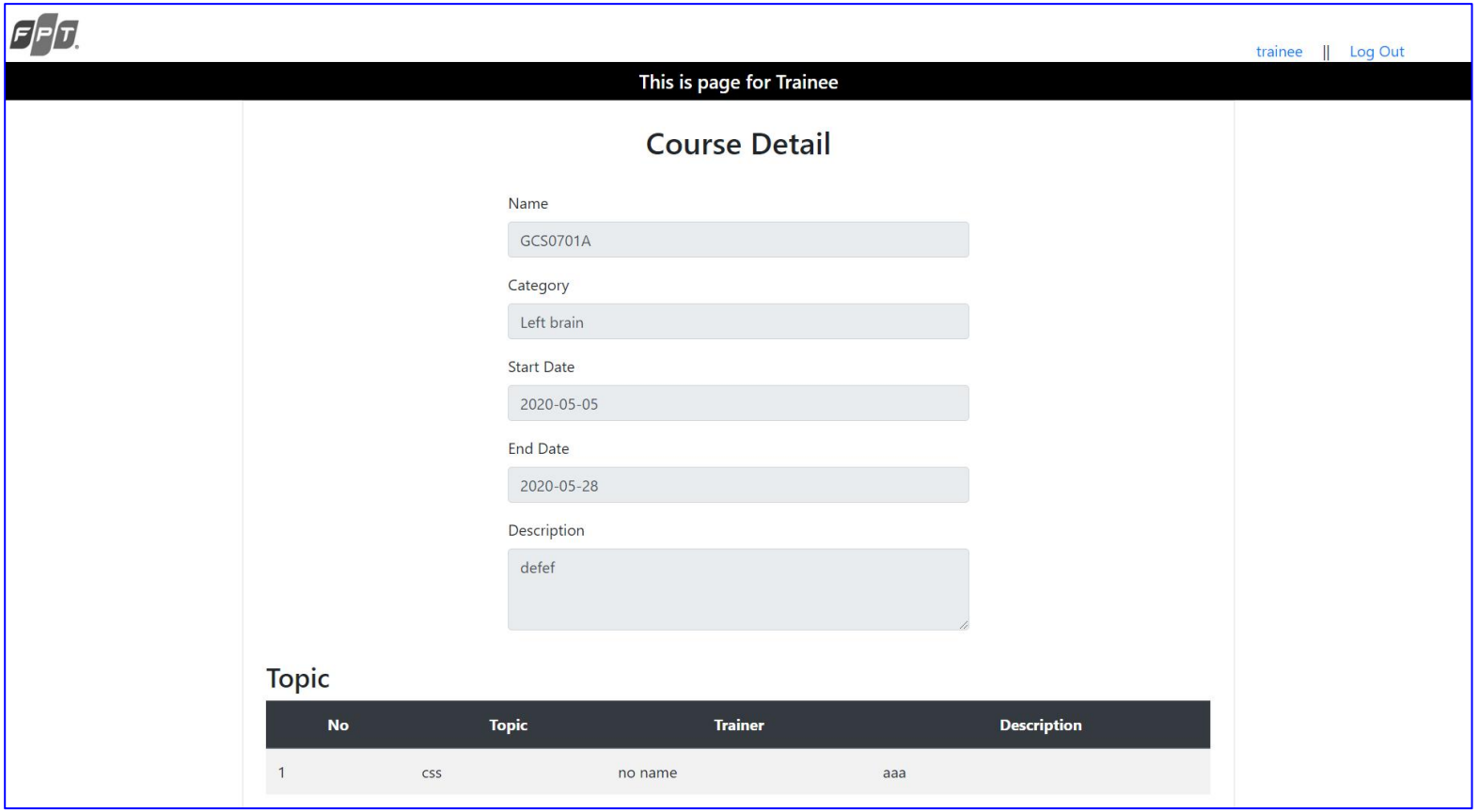

Figure 26 : The page show the detail of assigned course of login trainee

5. Review performance

5.1. Test plan

- **Admin**

No	Task	Expected result
1	Login as Admin	The application display websites for Admin after login with Admin account
2	Search account via username	The application display list of account with the appropriate account with username which have the key words in search bar
3	Edit username and password of an account	The username and password are modified after the action.
4	Delete account	The specific account will be completely remote from the application database
5	Log out	The application redirect to log in page and user can not go to other pages without re-login.

- **Training Staff**

No	Task	Expected result
1	Login as Training Staff	The application display websites for Training Staff after login with Training Staff account
2	Edit training staff profile	Training staff successfully edit his or her profile information
3	Show list of category	The category index page show the list of category existed in the system.
4	Search category via its name	The application display list of category with the appropriate name which have the key words in search bar
5	Add new category	New category will be added into the system.
6	Edit existing category information	Category information is modified after Training staff update it.
7	Delete category	Category is completely deleted from the application.
8	Remove course from category	Course will be removed from the course table which is display in the category edit page.
9	Show list of course	The course index page show the list of course existed in the system.
10	Search course via its name	The application display list of course with the appropriate name which have the key words in search bar

11	Add new course	New course will be added into the system.
12	Edit existing course information	Course information is modified after Training staff update it.
13	Delete course	Course is completely deleted from the application.
14	Assign Topic into course	The assigned topic will be display in the topic table in the course edit page
15	Assign Trainee into course	The assigned trainee will be display in the trainee table in the course edit page
16	Remove topic from course	Topic will be removed from the topic table which is display in the course edit page.
17	Remove trainee from course	Trainee will be removed from the trainee table which is display in the course edit page.
18	Show list of topic	The topic index page show the list of topic existed in the system.
19	Search topic via its name	The application display list of topic with the appropriate name which have the key words in search bar
20	Add new topic	New topic will be added into the system.
21	Edit existing topic	Topic information is modified after Training staff update it.

	information	
22	Delete topic	Topic is completely deleted from the application.
23	Assign Trainer into topic	The assigned trainer will be display in the trainer table in the topic edit page
24	Remove trainer from topic	Trainer will be removed from the trainer table which is display in the topic edit page.
25	Show list of trainer	The trainer index page show the list of trainer existed in the system.
26	Search trainer via its name	The application display list of trainer with the appropriate name which have the key words in search bar
27	Edit existing trainer information	Trainer information is modified after Training staff update it.
28	Show list of trainee	The trainee index page show the list of trainee existed in the system.
29	Search trainee via its name, account name, TOEIC score, and so on…	The application display list of trainee with the appropriate information which have the key words in search bar
30	Add new trainee account	New trainee account will be added into the system.

31	Edit existing trainee information	Trainee information is modified after Training staff update it.
32	Edit existing trainee account	Trainee account username and password is modified after Training staff update it.

- **Trainer**

No	Task	Expected result
1	Login as Trainer	The application display interface for Trainer after login with trainer account
2	Show list of topic	The application display list of topic which the login trainer is assigned to
3	Show list of course	The application display list of course which the login trainer is assigned to
4	Edit profile	The login trainer information is modified after he or she update it.
5	Show the detail of course	The application display the detail of a course and list of trainee who is assigned to it.

- **Trainee**

No	Task	Expected result
1	Login as Trainee	The application display interface for Trainee after login with trainee account
2	Show list of course	The application display list of course which the login trainee is assigned to
3	Show the profile	The application display the profile information of the login trainee.
4	Show the detail of course	The application display the detail of a course and list of topic which is assigned to it.

5.2. Test result

- **Admin**

No	Task	Result	Date	Error Description
1	Login as Admin	Successful	08/05/2020	
2	Search account via username	Successful	08/05/2020	

3	Edit username and password of an account	Successful	08/05/2020	
4	Delete account	Partial successful	08/05/2020	When the admin delete his or her own account, they still can use the application before they log out. The application should make them log out automatically.
5	Log out	Successful	08/05/2020	

- **Training Staff**

No	Task	Result	Date	Error Description
1	Login as Training Staff	Successful	08/05/2020	
2	Edit training staff profile	Successful	08/05/2020	
3	Show list of category	Successful	09/05/2020	
4	Search category via its name	Successful	09/05/2020	
5	Add new category	Successful	09/05/2020	
6	Edit existing category information	Successful	09/05/2020	
7	Delete category	Successful	09/05/2020	

8	Remove course from category	Successful	09/05/2020	
9	Show list of course	Successful	09/05/2020	
10	Search course via its name	Successful	09/05/2020	
11	Add new course	Successful	09/05/2020	
12	Edit existing course information	Successful	09/05/2020	
13	Delete course	Successful	09/05/2020	
14	Assign Topic into course	Successful	09/05/2020	
15	Assign Trainee into course	Successful	09/05/2020	
16	Remove topic from course	Successful	09/05/2020	
17	Remove trainee from course	Successful	09/05/2020	
18	Show list of topic	Successful	09/05/2020	
19	Search topic via its name	Successful	09/05/2020	
20	Add new topic	Successful	09/05/2020	
21	Edit existing topic information	Successful	09/05/2020	
22	Delete topic	Successful	09/05/2020	

23	Assign Trainer into topic	Successful	09/05/2020	
24	Remove trainer from topic	Successful	09/05/2020	
25	Show list of trainer	Successful	09/05/2020	
26	Search trainer via its name	Successful	09/05/2020	
27	Edit existing trainer information	Successful	09/05/2020	
28	Show list of trainee	Successful	09/05/2020	
29	Search trainee via its name, account name, TOEIC score, and so on…	Successful	09/05/2020	
30	Add new trainee account	Successful	09/05/2020	
31	Edit existing trainee information	Successful	09/05/2020	
32	Edit existing trainee account	Successful	09/05/2020	

- **Trainer**

No	Task	Result	Date	Error Description
1	Login as Trainer	Successful	08/05/2020	

2	Show list of topic	Successful	08/05/2020	
3	Show list of course	Successful	08/05/2020	
4	Edit profile	Successful	08/05/2020	
5	Show the detail of course	Successful	08/05/2020	

- **Trainee**

No	Task	Result	Date	Error Description
1	Login as Trainee	Successful	08/05/2020	
2	Show list of course	Successful	08/05/2020	
3	Show the profile	Successful	08/05/2020	
4	Show the detail of course	successful	08/05/2020	

References

[1]. Entityframeworktutorial.net. 2020. *Fluent API In Entity Framework 6*. [online] Available at: <https://www.entityframeworktutorial.net/code-first/fluent-api-in-code-first.aspx> [Accessed 9 May 2020].

[2]. Entityframeworktutorial.net. 2020. *Configure One-To-One Relationship In Code First Entity Framework*. [online] Available at: <https://www.entityframeworktutorial.net/code-first/configure-one-to-one-relationship-in-code-first.aspx> [Accessed 9 May 2020].

[3]. Entityframeworktutorial.net. 2020. *Configure One-To-Many Relationship In Entity Framework 6*. [online] Available at: <https://www.entityframeworktutorial.net/code-first/configure-one-to-many-relationship-in-code-first.aspx> [Accessed 9 May 2020].

[4]. Entityframeworktutorial.net. 2020. *Configure Many-To-Many Relationship In Code First*. [online] Available at: <https://www.entityframeworktutorial.net/code-first/configure-many-to-many-relationship-in-code-first.aspx> [Accessed 9 May 2020].

[5]. Entityframeworktutorial.net. 2020. *Cascade Delete In Entity Framework 6*. [online] Available at: <https://www.entityframeworktutorial.net/code-first/cascade-delete-in-code-first.aspx> [Accessed 9 May 2020].

[6]. Mark Otto, a., 2020. *Tables*. [online] Getbootstrap.com. Available at: <https://getbootstrap.com/docs/4.4/content/tables/> [Accessed 10 May 2020].

[7]. Mcleod, S., 2020. *Questionnaire: : Definition, Examples, Design And Types | Simply Psychology*. [online] Simplypsychology.org. Available at: <https://www.simplypsychology.org/questionnaires.html> [Accessed 10 May 2020].

[8]. Mark Otto, a., 2020. *Grid System*. [online] Getbootstrap.com. Available at: <https://getbootstrap.com/docs/4.1/layout/grid/> [Accessed 10 May 2020].

[9]. application?, H., Gaotingwe, T., Enamno, J., Woodward, J., Maccaferri, L. and Sharma, M., 2020. *How To Use Sessions In An ASP.NET MVC 4 Application?*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/14138872/how-to-use-sessions-in-an-asp-net-mvc-4-application> [Accessed 10 May 2020].

[10]. js.foundation, J., 2020. *Jquery.Ajax()* | *Jquery API Documentation*. [online] Api.jquery.com. Available at: <https://api.jquery.com/jquery.ajax/> [Accessed 10 May 2020].