



**Dharmsinh Desai University, Nadiad**  
**Faculty of Technology**  
**Department of Computer Engineering**

**B. Tech. CE Semester – IV**

**Subject: SEPP**

**Project title: Online Educational System**

**By:**

**1) Kairav R. Gamit , Roll No:CE035 , id : 19CEUTG008**

**2) Karnav R. Gamit , Roll No:CE036 , id: 19CEUTG091**

**Guided by: Prof. Brijesh S. Bhatt**

**Prof. Pinkal C. Chauhan**

**Prof. Jigar M. Pandya**

**DHARMSINH DESAI UNIVERSITY**  
**NADIAD-387001, GUJARAT**



**CERTIFICATE**

This is to certify that the project entitled as “**Online Educational System**” is a bonafide report of the work carried out by

**1) Mr. Kairav Rajanikant Gamit, Student ID No: 19CEUTG008**

**2) Mr. Karnav Rajanikant Gamit, Student ID No: 19CEUTG091**

of Department of Computer Engineering, semester IV, under the guidance and supervision of **Prof. Brijesh S. Bhatt** for the subject System Design Practice during the academic year 2020-2021.

Project Guide  
Professor  
Prof. Brijesh S. Bhatt  
Department of Computer  
Engineering,  
Faculty of Technology,  
Dharmsinh Desai University,  
Nadiad  
Date: 31/03/2021

Head of the Department  
Dr. C.K Bhensdadia  
Prof. & Head,  
Department of Computer  
Engineering,  
Faculty of Technology,  
Dharmsinh Desai University,  
Nadiad  
Date: 31/03/2021

# **Contents:**

1. Abstract.....	4
2. Introduction .....	5
3. Software Requirement Specifications .....	6
4. Design	
I) Use Case diagram .....	10
II) Class diagram.....	11
III) Sequence diagram .....	12
IV) Activity diagram .....	13
V) Data Flow diagram .....	14
VI)Major Functionality .....	18
5. Workflow .....	23
6. Conclusion.....	26
7. Bibliography .....	27

## **Abstract**

Nowadays during this pandemic time , all professions are doing work online , and also teaching side is affected and students are learning online , so we can build a website where students can learn courses at their place at their time.so some problems or function are as follow:

- 1) If there is no any type of website or platform for it , then firstly it creates a problem for those people who are living in a remote place. Because it is hard for them to visit the institute personally.
- 2) There can be a case of fraud with them by using the name of the institute due to indirect contact with it.
- 3) Most useful feature is that they can contact directly to the institute for any query or confusion with any interface of third party.
- 4) They can easily access any material of the course from any where.
- 5) In the absence of online website it is difficult to communicate with the institute for any emergency update or for any needed information about a particular course.
- 6) Also they can easily pay fee or expenses of institute without any problem.

### **Description:**

- 1) This site have mainly the feature of the details about the different courses which are available in the institute.
- 2) Online lectures and videos are also available for the students.
- 3) All the basic structure of the institute about their courses is available on site.
- 4) FAQ section or Query section is also a useful feature for the user to solve their doubt or query regarding institute or course.
- 5) Material for a particular course can be easily provide in the form of pdf of books. So there is no any problem for lack of material for any student
- 6) Unique login id is provided to every user, so no any other user out of the institute can access to the resources of the institute.

# **Introduction**

Online Educational System in which there are mainly two users Admin and students. By this System students can learn by their comfort .Any student can signup and able to find their course in which they are interested. Which will be provided by Admin. When any student visits the website he has to sign up first.

After login the students have to pay for enrolling in any course. Admin can upload or remove course and user.

## **Technologies/tools used :**

- Platform used: Visual Studio Code
- Technology: Django, sqlite3

# **Software Requirement Specifications:**

## **Online Educational System**

### **❖ Login**

#### **R.1 :**

- **Description:** In this , user will login using his/her username and password.

If user haven't signed up then he/she has to sign up first.

#### **R.1.1:**

- **Description:** If user haven't signed up before user will create an account by providing his/her details.
  - **Input :** User will provide his/her needed information in the form.
  - **Output:** User will get message , confirming the creation of account successfully .
- **Process :** Entered username and password will get verified.

#### **R.1.2 :**

- If user have already signed up before successfully then user has to enter username and password to login.

### **❖ Home Page**

- **State :** User should be logged in through his/her account successfully.
- **Description :** User will see this home screen after the login.

#### **R.1 :**

- **Description :** User will lead to some options of the website available on the home page.
- **Input :** User will select one option given on the page.
- **Output:** It will lead to the detailed page for that option. Where user can perform further tasks.

### **R.1.1 : For tutorial**

- **Input** : User will select appropriate tutorial which he/she wants to access.
- **Output** : Selected tutorial will be displayed to user.

### **R.1.2 : Help**

- **Input** : User can write to this section for which he/she needs help.
- **Output** : User will be lead to appropriate section where his confusion can be solved.

### **R.1.3 Profile management**

- **Input** : User will select option given to him/her to manage profile.  
Then before going further user has to provide current account password.  
If the password is correct user can make changes.
- **Output** : Appropriate changes will be shown or done based on the changes user has done.

#### **R.1.3.1 Change password**

- **Input** : User has to enter current password before changing it to new one.
- **Output** : Confirmation message will shown and password will be changed.
- **Process** : If user has entered correct password only then user can change password.  
After that user has to enter new password two times.

#### **R.1.3.2 Change personal Info**

- **Input** : User will enter (select) which information he/she wants to change. Based on it user has to enter new information.
- **Output** : Updated information can be seen in the user's profile after confirmation message.

## **❖ Course**

### **R.1:**

- **Description** : User can see details of the courses available on the site and after that user will proceed further for particular course.

- **Input** : User will select course from the list of the course given in the options.
- **Output** : User will be lead to payment section.

#### **R.1.1 : Payment**

- **Description** : User will pay particular fees for selected course.
- **Input** : User will provide needed information for payment.
- **Output** : Generated copy of Receipt will be provided to user and user will be enrolled in selected course.
- **Process** : If transaction get successful then user will be added to the course.

### ❖ Feedback

#### **R.1**

- **Description:** In this module , user can rate the course as well he can give his suggestions .
- **Input** : user can rate and provide text description of his review.
- **Output:** we will show a text that users response has recorded.

### ❖ Student

#### **R.1:**

- **State:** User must have logged in to website.
- **Description** : Used for maintaining students details. Student can maintain their profile and update/change their profile. As well we will display in which course user is in and when is the end date of course.
- **Input** : user will select edit profile and can edit his/her info and in courses user can get details about his/her enrolled course.



- User can select material section and can access the related material. And also have to see video lectures provided.
- **Output** : we will ask for confirmation and after that we will update it in database.

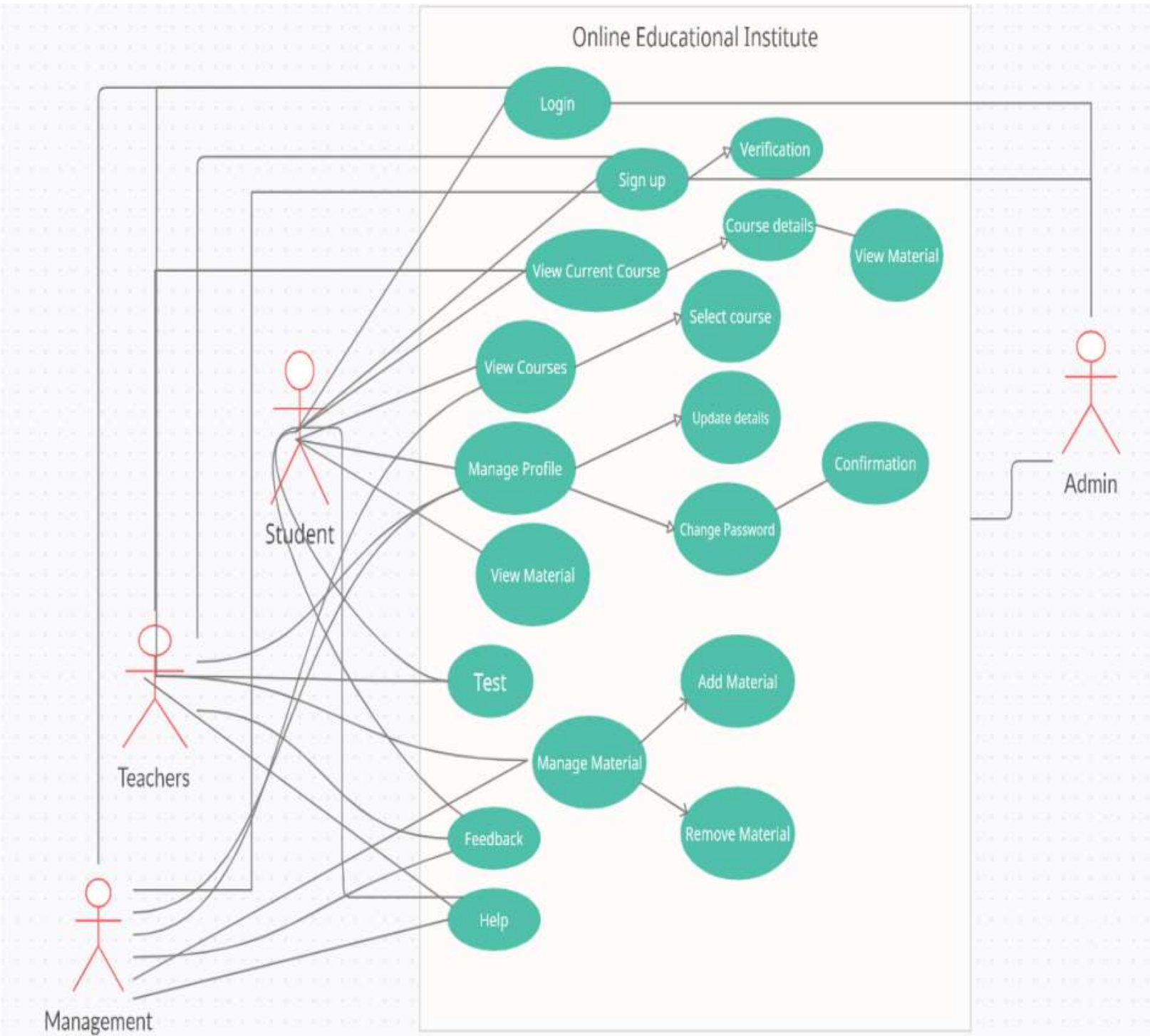
## ❖ Teacher

### R.1 :

- **State** : User must have logged in before going to further steps.
- **Description** : user can upload documents and materials to the material section of course he/she teaches. Teacher also can mail to students via website and can contact them. Teacher also can upload test .
- Also can update his/her video lectures in particular course. And can see how many students have seen.
- **Input** : User will select material section and can update it . and also update on database.
- **Output** : we will display that updating has been done.

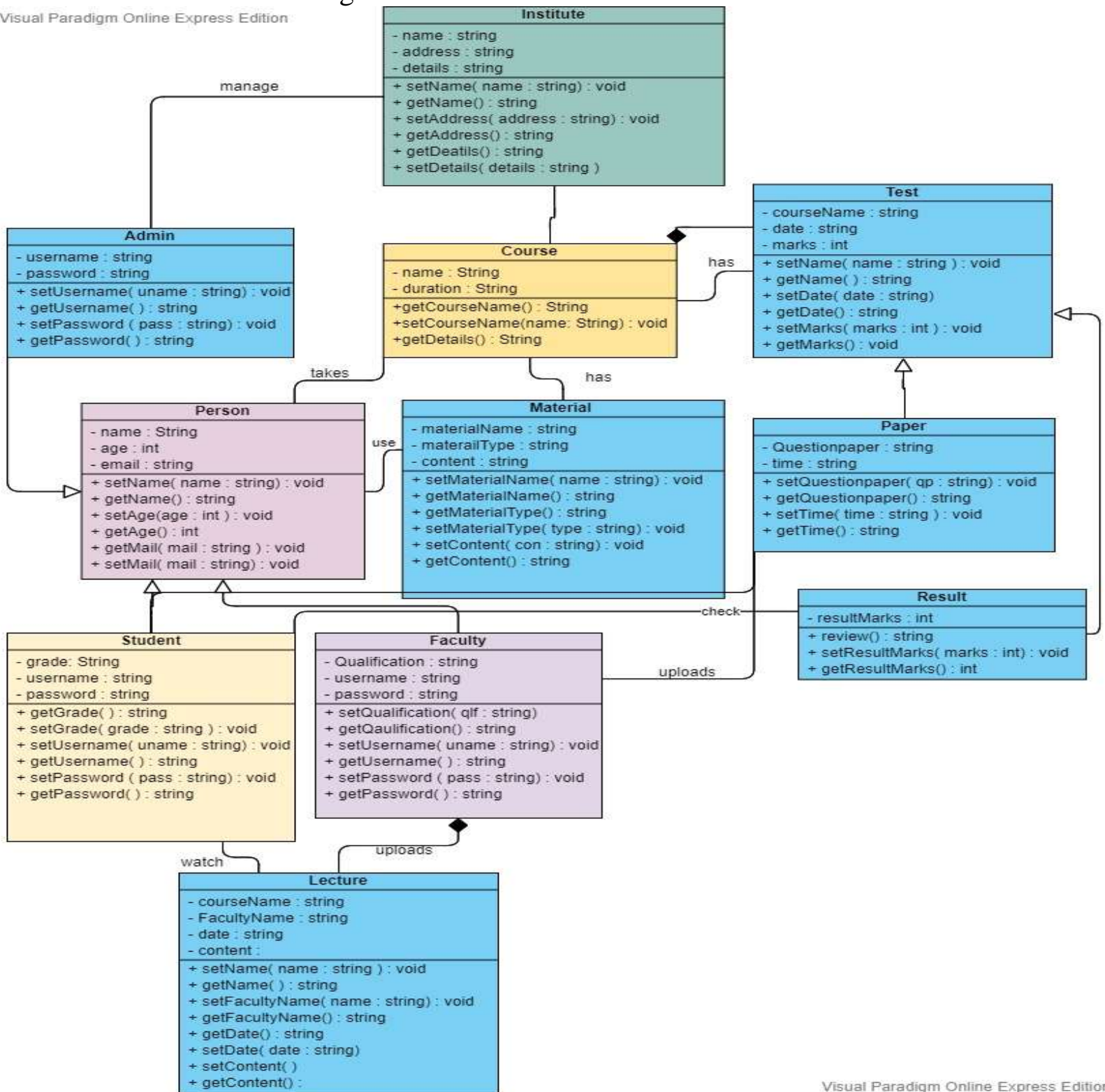
## Design:

- Use Case diagram



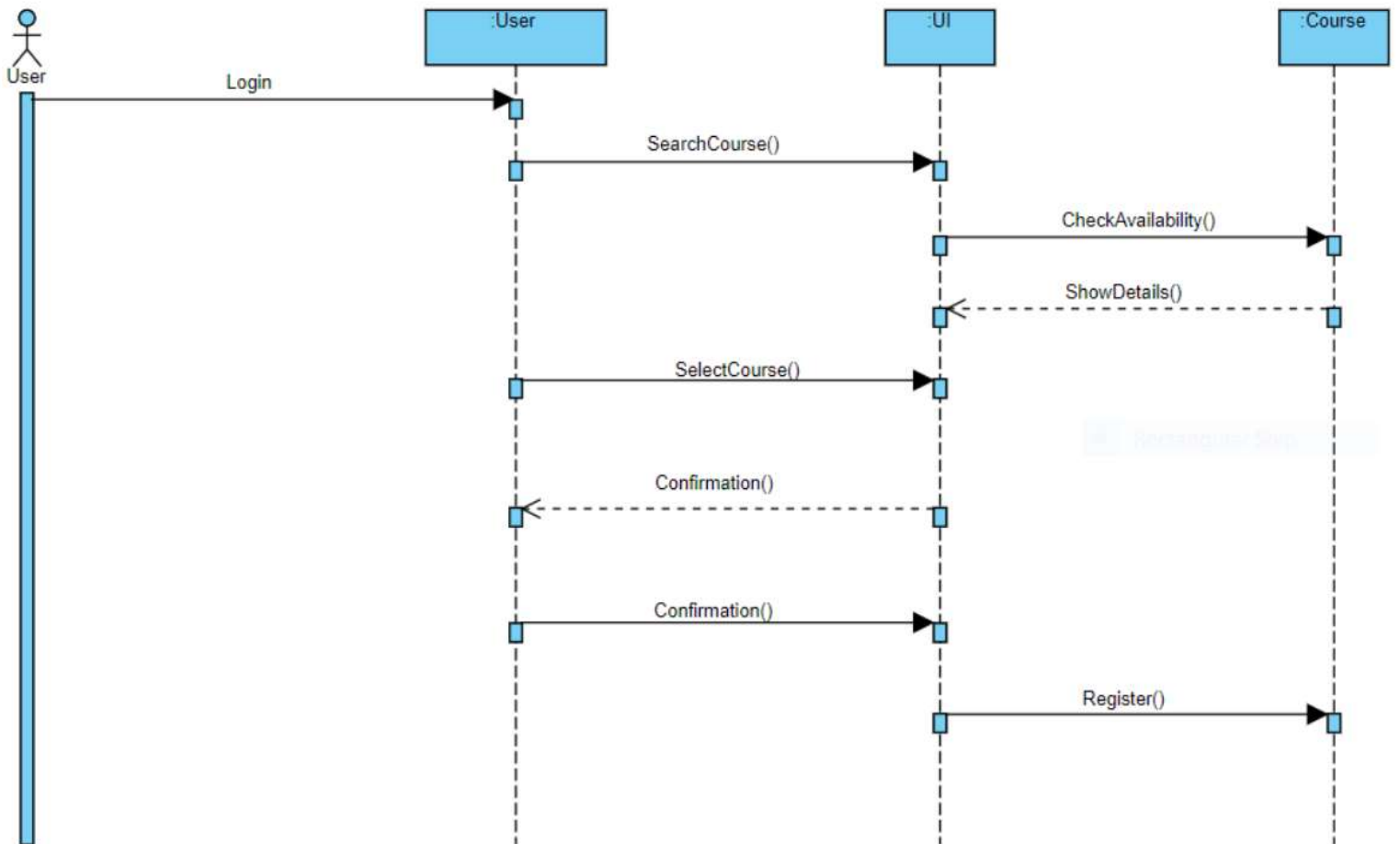
## • Class Diagram

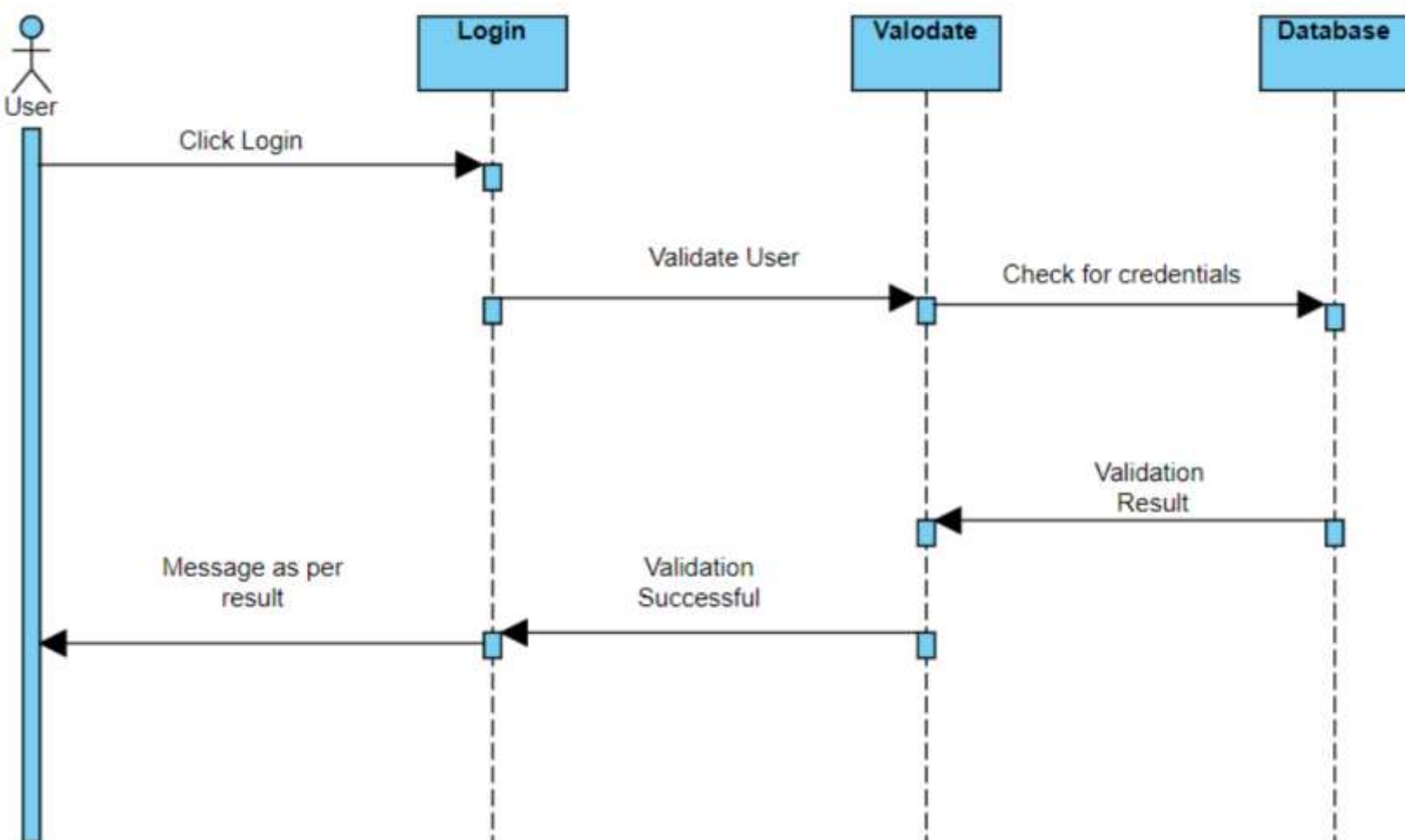
Visual Paradigm Online Express Edition



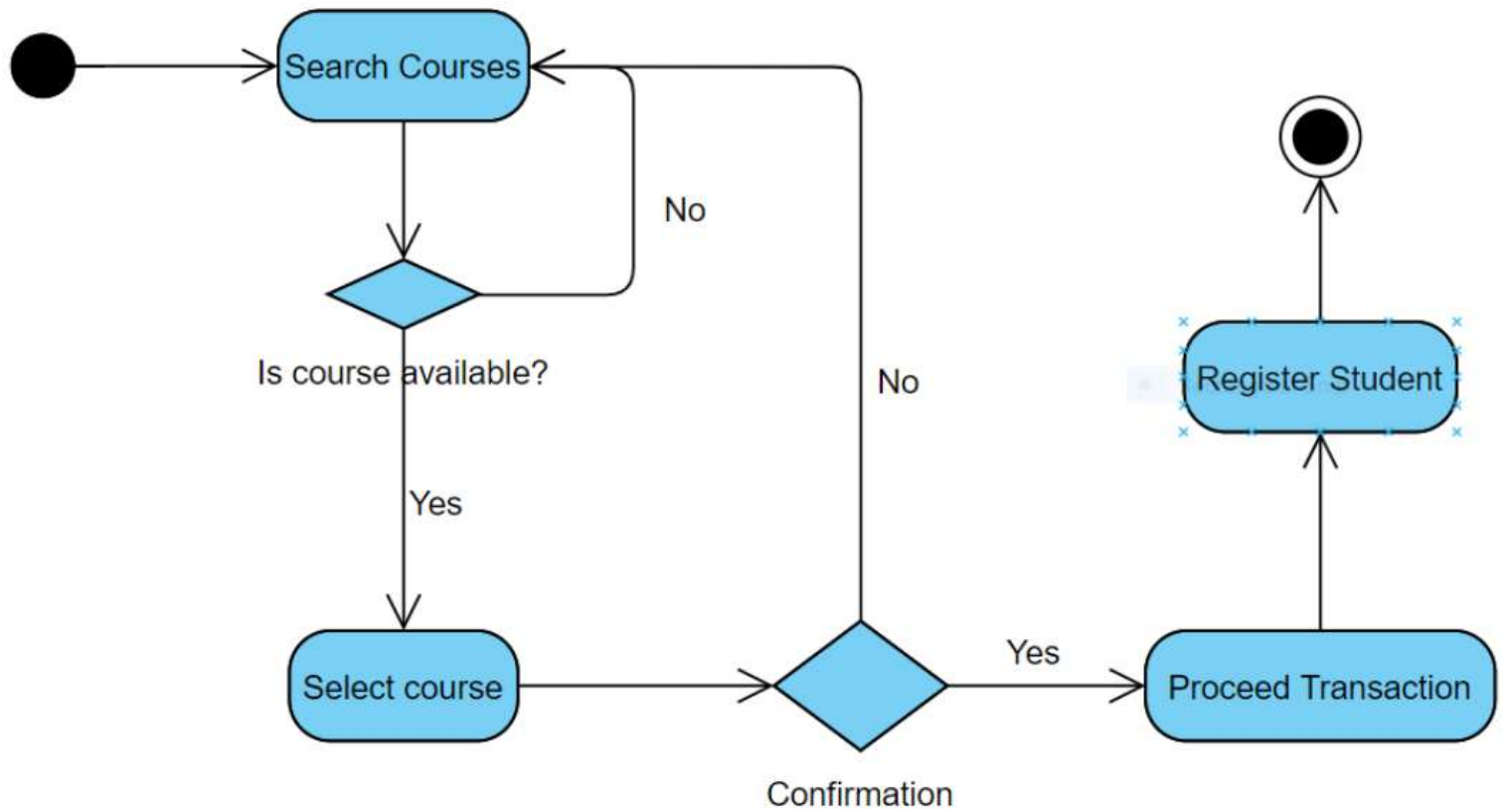
Visual Paradigm Online Express Edition

- Sequence Diagram

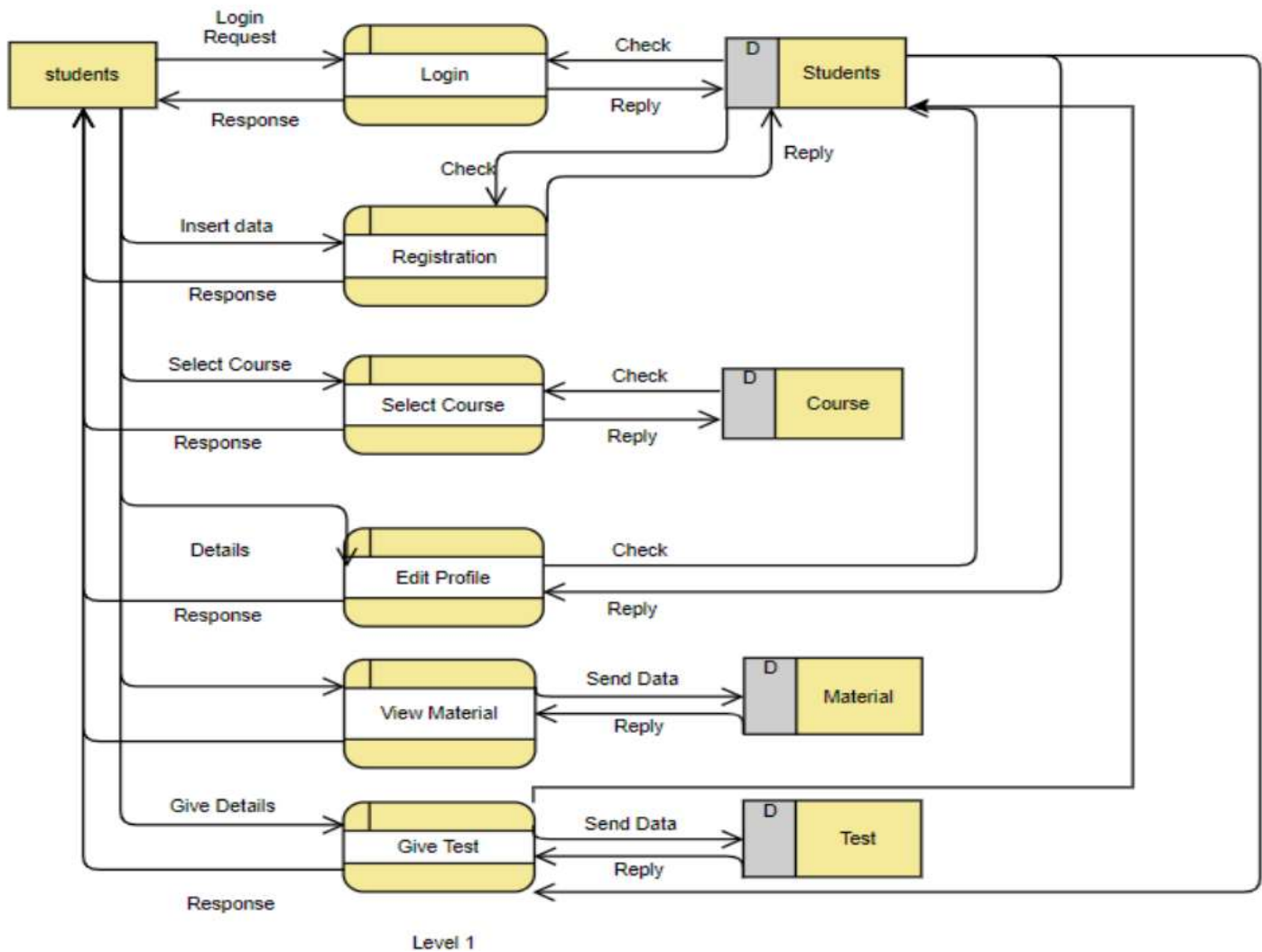




- Activity Diagram

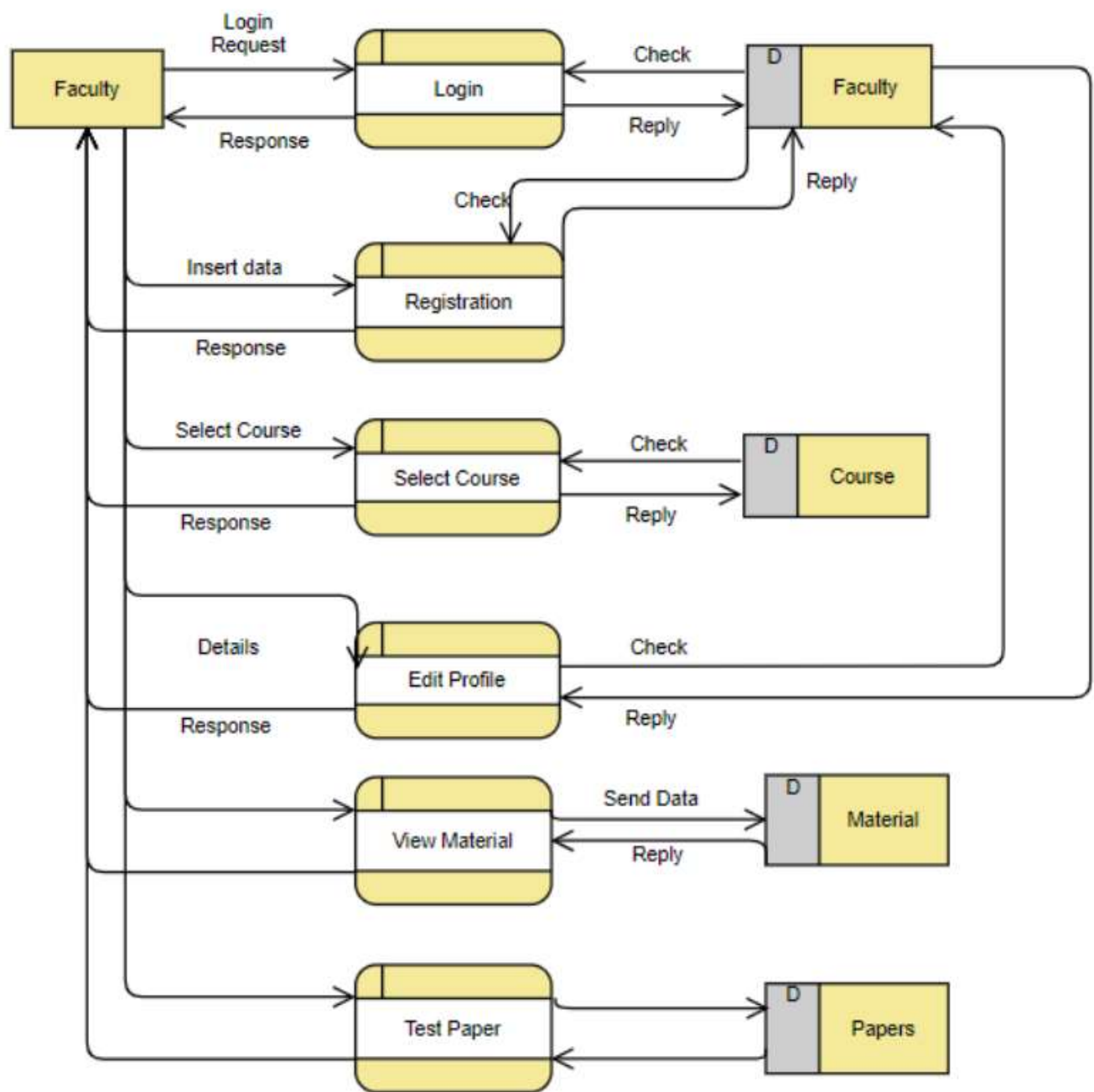


- Data Flow Diagram



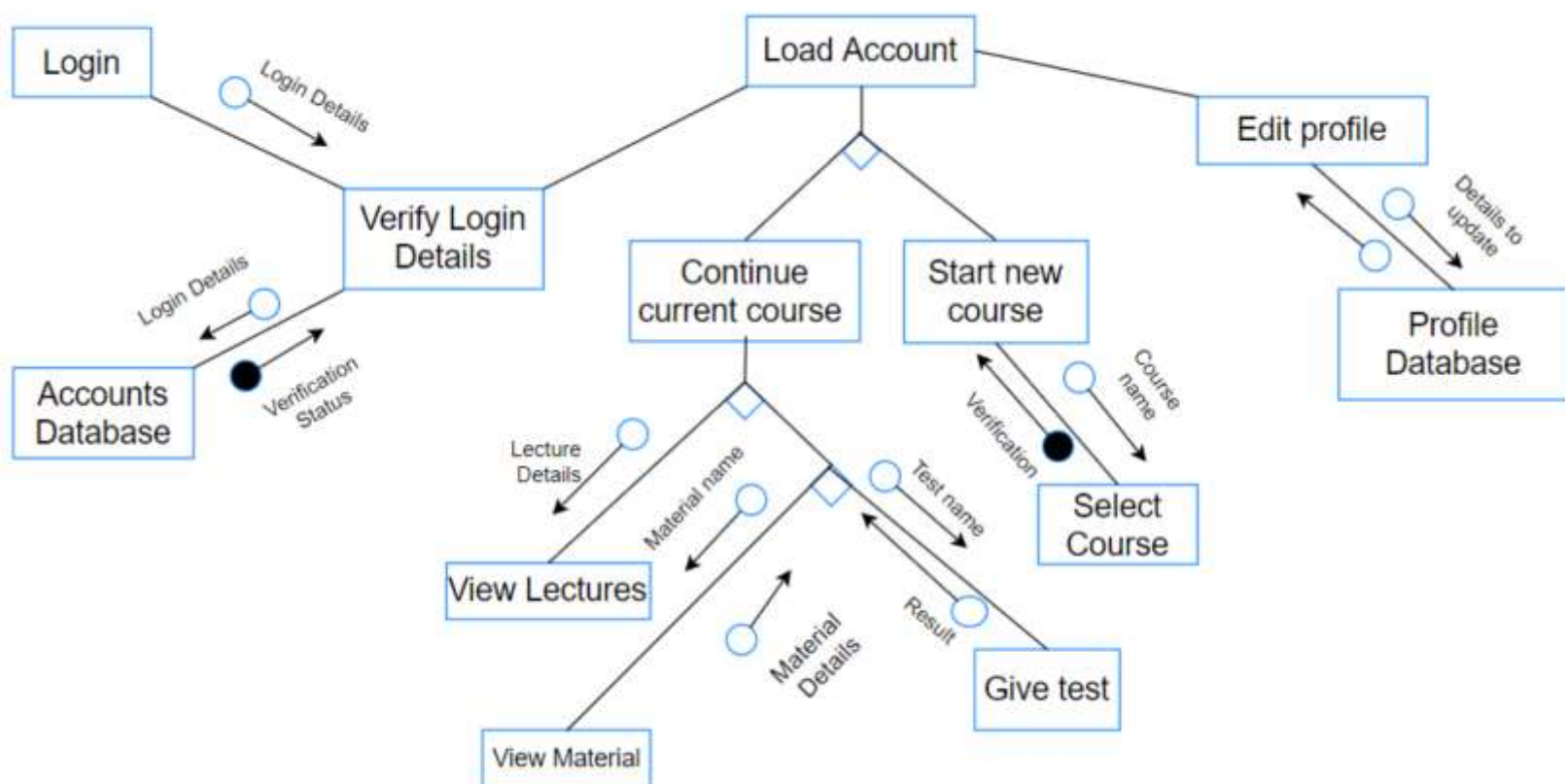


# Faculty Side DFD



Level 1





## 1. Major Functions prototypes

### 1. Home :

Home function is just simply for the view of home page. In which it shows the available courses.

```
def home(request):  
    courses = Course.objects.all()  
    print(courses)  
    return render(request, 'home.html', {'courses': courses})
```

### 2. Course page function :

This function will show the page for the particular course selected.

```
def coursePage(request, slug):  
    course = Course.objects.get(slug = slug)  
    serial_number = request.GET.get('lecture')  
    videos = course.video_set.all().order_by("serial_number")  
    if serial_number is None:  
        serial_number = 1  
  
    video = Video.objects.get(serial_number = serial_number , course = course)  
  
    if (video.is_preview is False):  
        if request.user.is_authenticated is False:  
            return redirect("login")  
        else:  
            user = request.user  
            try:  
                user_course = UserCourse.objects.get(user = user , course = course)  
            except:  
                return redirect("checkout", slug = course.slug )  
  
    context = {  
        "course": course ,  
        "video" : video ,  
        "videos" : videos  
    }  
    return render(request, 'course_page.html', context )
```

### 3. Signup function:

In this function user is being handled for the first time. Through this he/she will register by providing some details in the signup form.

Note : Here we have used class view instead of a simple function.

```
class SignupView(View):
    def get(self, request):
        form = RegistrationForm()
        return render(request, "signup.html" , {'form' : form})

    def post(self, request):
        form = RegistrationForm(request.POST)

        if(form.is_valid()):
            user = form.save()
            if(user is not None):
                return redirect("login")

        return render(request, "signup.html" , {'form' : form})
```

#### 4. Login :

In this class view user will get simply logged in by providing his/her email and appropriate password in the login form.

```
class LoginView(View):
    def get(self, request):
        form = LoginForm()
        context = { "form" : form }

        return render(request, "login.html", context)

    def post(self, request):
        form = LoginForm( request = request, data = request.POST)
        context = { "form" : form }
        if(form.is_valid()):
            return redirect("home")

        return render(request, "login.html" , context)
```

#### 5. Logout :

It is just a simple function for logging out. After logging out it will redirect user to home page.

```
def signout(request):
    logout(request)
    return redirect("home")
```

## 6. Checkout function:

This is a function created for checkout of the course while enrolling in the course.

```
def checkout(request, slug):
    course = Course.objects.get(slug = slug)
    user = None

    if not request.user.is_authenticated:
        return redirect("login")

    user = request.user
    action = request.GET.get('action')
    order = None
    payment = None
    error = None

    if action == 'create_payment':
        try:
            user_course = UserCourse.objects.get(user = user , course = course)
            error = "You are already inrolled in course!"
        except:
            pass

        if error is None:
            amount = int((course.price) * 100)
            currency = "INR"
            notes = {
                "email" : user.email
            }
            receipt = f"MyInstitute-{int(time())}"
            order = client.order.create({'receipt' : receipt , 'notes': notes , 'amount' : amount , 'currency' : currency})

            payment = Payment()
            payment.user = user
            payment.course = course
            payment.order_id = order.get('id')
            payment.save()

        context = {
            "course" : course ,
            "order" : order ,
            "payment" : payment ,
            "user" : user ,
            "error" : error ,
        }
    return render(request, 'check_out.html', context)
```

In this function it will first check that if user is logged in or not , if user is not logged in before checkout user will be redirected to login page for proceeding further. If user is already logged in then it is checking that if user is already enrolled in the course or not , if so then an error message will be shown to user otherwise it will check further.

If all conditions are clear then a order object will be created and for that order id payment object will be created and will be saved.

## 7. VerifyPayment

In this payment for selected particular course will be get verified.

Here 'razorpay' is being used for this purpose. If user's payment get verified then payment status for that user will be changed to true , false otherwise. And user will get access to explore the course.

```
@csrf_exempt
def verify_payment(request):
    if request.method == "POST":
        data = request.POST
        context = {}
        print(data)
        try:
            client.utility.verify_payment_signature(data)
            razorpay_order_id = data['razorpay_order_id']
            razorpay_payment_id = data['razorpay_payment_id']

            payment = Payment.objects.get(order_id = razorpay_order_id)
            payment.payment_id = razorpay_payment_id
            payment.status = True

            userCourse = UserCourse(user = payment.user , course = payment.course)
            userCourse.save()

            print("Usercourse" , userCourse.id)

            payment.user_course = userCourse
            payment.save()

            return render(request, "my_courses.html", context)
        except:
            return HttpResponse("Invalid Payment Details")
```

## 8. My courses function :

In this function it will show the enrolled courses of user.

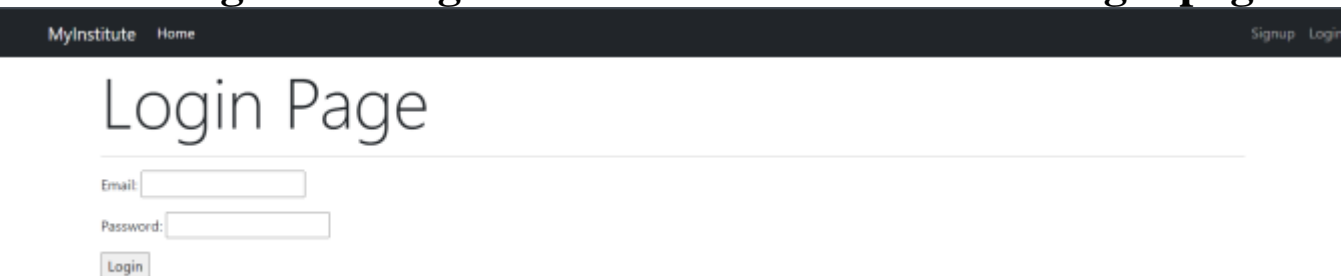
```
@login_required(login_url="login")
def mycourses(request):
    user = request.user
    user_courses = UserCourse.objects.filter(user = user)
    context = {
        'user_courses' : user_courses
    }
    return render(request, "my_courses.html" , context)
```

# WORKFLOW:

## Home page:



## Without login if user goes to enroll user redirects to login page:



## After login :

[MyInstitute](#) [Home](#) [My Courses](#) [Logout](#)



**C Language**  
Famous and basic programming language  
₹ 999

[Enroll Now](#) [Show More](#)



**C++**  
This language is called updated version of C language  
₹ 999

[Enroll Now](#) [Show More](#)




**Java**  
The most object oriented programming language  
₹ 1499

[Enroll Now](#) [Show More](#)

## After that clicking on enroll now:

[MyInstitute](#) [Home](#) [My Courses](#) [Logout](#)



**C Language**  
famous and basic programming language  
₹ 999

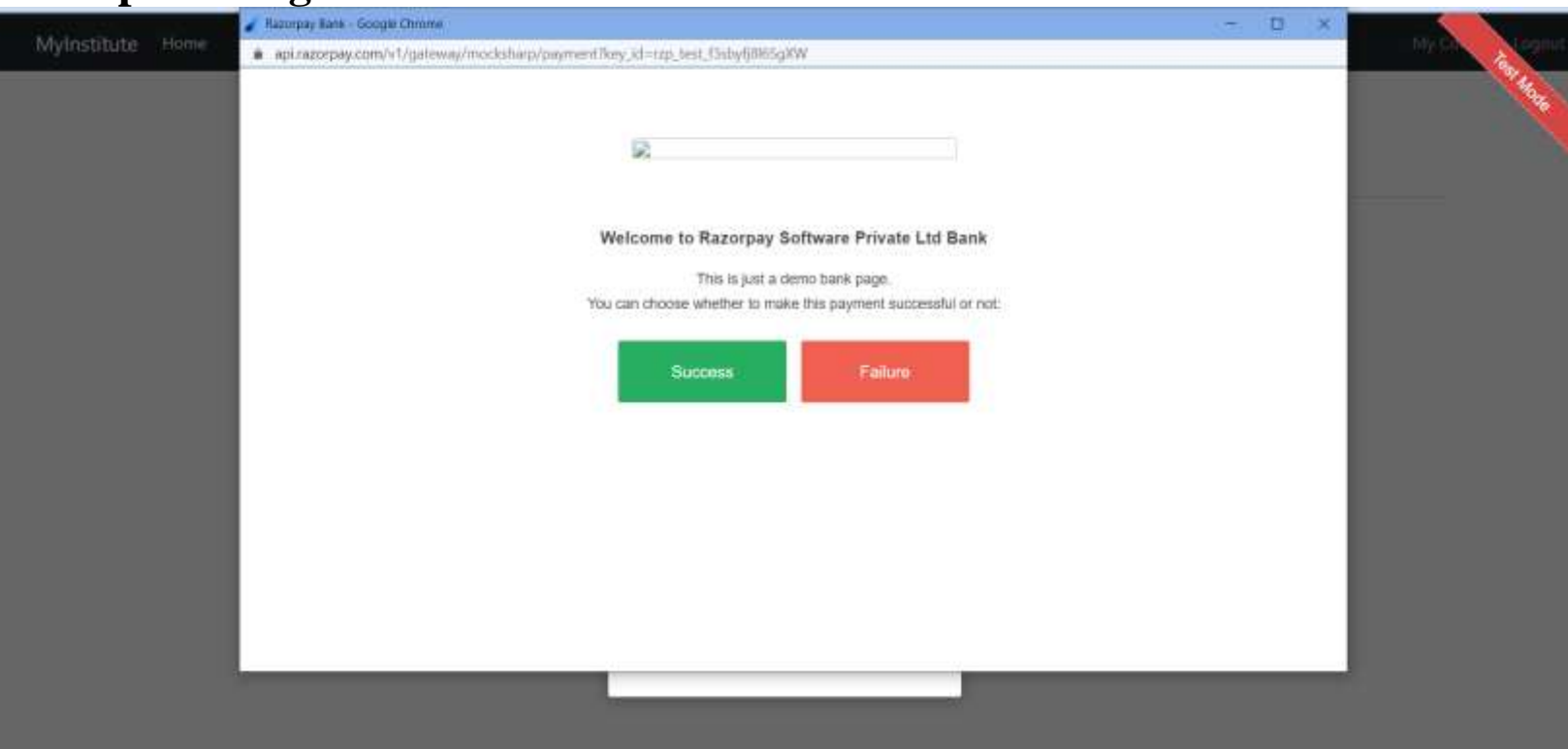
### C Language

Payable Amount : ₹ 999

[Continue to payment](#)



**After providing details :**



**After enrolling in my courses:**



## **Conclusion:**

Hence-forth in this project we have successfully implemented the Admin-side & student side functionality. Admin will add the courses details and. And the user can buy the courses per their choice . As well uploaded material videos can be viewed and student can learn the concepts and study online.

## **LIMITATIONS:**

- 1) Any of the payment modes does not actually link the bank account to the entered details of customer and the amount is not deducted from the account.
- 2) Project is for learning purpose so there is no actual institute for this website.

## **Future Extension:**

To take over the limitations we are planning this future extension in our system.

- 1) Some more detailed feature can be implemented for future extension like download pdfs for courses.
- 2) User can have a separate profile for interfering .

# **Bibliography**

## **References/resources used for developing project:**

- <https://stackoverflow.com/>
- <https://www.geeksforgeeks.org/>
- [https://www.youtube.com/results?search\\_query=telusko+django](https://www.youtube.com/results?search_query=telusko+django)