A PROJECT REPORT

ON

STUDY PORTAL

By

ANTIMA CHAUHAN

Towards the Partial Fulfilment of the

Bachelor of Computer Application SEM-VI

SHAILENDRA EDUCATION SOCIETY'S

COMPUTER CENTER (SESCOM)

Mumbai 400 068

Tilak Maharashtra Vidyapeeth, Pune

Department of Computer Science

2021-2022

CERTIFICATE

This is to certify that the project

STUDY PORTAL

Has been satisfactorily completed by

ANTIMA CHAUHAN

Towards the Partial Fulfillment of the 'Bachelor of Computer Application SEM-VI', For the Academic 2021-2022 At Shailendra Education Society's Computer Center (SESCOM), Mumbai 400 068. Tilak Maharashtra Vidyapeeth, Pune (Department of Computer Science), and it is approved.

Project Guide **Examiner** **Head of Department**

SESCOM-Mumbai 68

ACKNOWLEDGEMENT

With immense please we are presenting “Study Portal” Project report as part of the curriculum of ‘Bachelor of Computer Application’. We wish to thank all the people who gave us unending support.

We express our profound thanks to our head of department Mr. Bhushan Pimple, project guide Prof. R. H. Gohel and project in-charges and all those who have indirectly guided and helped us in the preparation of this project.

ANTIMA CHAUHAN (PRN NO. :04419002743)

INDEX

Chapter 1: Introduction page No.

Chapter 2: System Analysis

Chapter 3: System Requirement Specifications

Chapter 4: System Design

Chapter 5: Screenshots 35-45

Chapter 6: Implementation Details

Chapter 7: Testing

Chapter 8: Conclusion

Chapter 1: Introduction

Introduction to Study Portal:

Study portal is a web-based application developed in python (Django Framework) that helps to make study easier by providing a managed environment containing many features which are useful in the study as well as these features (like to-do, translator) can be used for our daily life activities. Users can take benefit of these features by simply registering themselves to the portal.

Study Portal provide an environment where users can participate in many activities like notes sharing, discussion Forum and take advantages of other features like notes creating/ managing, to-do app for planning the day, language translator, book search functionality, and pdf reader to listen to E-book.

The aim of the system is to provide study tools in one place so that user can save their time instead of Wander to find these things on different places.

It is easy to use with a user-friendly GUI it can be used by any technical or non-technical person. This portal provides study related features in one place, in very easy and secure manner where after registration all user have their own dashboard.

It is not intended to a particular organization this project going to develop generic software, which can be used by any study organization.

1.1 Introduction to all features:

ToDo:

User can plan their day by creating a to-do list for their day. This will help to improve time management as well as workflow, after completing the task user can mark the task as completed or delete it.

In this section, the user can create delete, and update the task.

Notes Creating:

Through the notes creating process users can create, update, download, and delete notes according to their needs.

In the notes creating section user can use many tool (like "Bold","Italic","heading","color", "style", "format" and many more) to create notes. Created notes are stored permanently until deleted. Notes, created by a user can only access by that user until the user does not share them with others

Notes Sharing:

In the notes sharing section user can upload, download and view the study notes according to their branch. The user can also edit the notes uploaded by him.

User can upload notes in the pdf, doc, ppt, zip, or txt file format. Users can search notes via subject, and title. Admin can accept or reject notes uploaded request. notes can be published only after accepted by the admin.

Online Discussion Forum:

The study portal contains a discussion forum section where user can share their thoughts, ideas, ask questions and help others by posting answers. The main purpose of the discussion forum is to provide a platform where user can ask question related to their fields and get answer easily. This forum will act as a healthy questionnaire which helps to give and get effective solutions with best of their experience

Here users can join the forum after coming on their dashboard.

Find Books:

find book section allows users to find their desired book online from the portal by simply entering the book name he will get a list of all books related his search.

Language Translator:

understanding a language is most important thing in our daily life because everything comes with communication.

our language translator will help user to understand a language by translating it into their desired language. Language translator allows user to type or paste text in input box, select desired language to translate text and user also able to download translated voice into their local system.

It is helpful for all user especially students can take benefits of this feature .it can Help students to understand the connection between languages and explores the potential of both of them. It is a necessary and natural activity, because in many common places foreign words are met and they need to be decoded.

PDF reader:

There are many situations comes in our daily life where we are facing language barrier to read a book. but now in my system I provide a pdf reader feature where user can listen their book pdf into their desired language by simply importing pdf and selecting their desired language. And user is also able to download translated audio book into their local system

1.2 Description

Study Portal developed will help to save time and make study easier for all by providing study tools in a single place. In this project two major modules that have been provided are administrator module and user module.

Admin Module:

The administrator is an authorized person who will keep track of all the uploaded notes and manage users as well through the admin panel. The Admin is a person who will have the authority to manage users and uploaded notes as well

Admin-functionality:

- Manage (View/Delete) user
- Manage(views/Delete) Notes
- Accept or reject Notes
- Manage (View/Edit/Delete) Post and reply.
- Change password and update profile

User Module:

All users have unique usernames and passwords to access the system. after login into the system, users can see their records and take benefit of all the features. and if the user forgets his login password he can reset his password throw email. The user should have a unique email id as well.

User-functionality:

- Create, update download and delete notes,
- Post/Edit questions and answer on discussion forum,
- Create to-do
- Change password and update profile,
- Upload, download, and search notes
- Delete notes uploaded by himself
- Search books online
- Translate language
- Listen pdf audio

1.3 Study Portal Objectives:

The objective of the system is to provide facilities in one place which are useful in the study so that user can save their time instead of Wander to find these things in different places. Portal design with user-friendly GUI which can be used by any technical or non-technical person.

The system is designed by keeping in mind the requirement of a user (who can be students and teachers or a person who is in study) as well as a study organization. After registration, each user has their own dashboard where user can contribute to many activities like notes sharing, discussion forum, and take advantage of other features like notes creating, work plans for the day, translating languages, and using a pdf reader to listen his E-book.

In this project, two major modules that have been provided are the administrator module and the user module. User can create their accounts and start using all 7 features, however, the user does not have the authority to delete the records of the website, This authority vests only with administrator. admin can manage user posts, replies, uploaded notes as well as user data.

Chapter 2: System Analysis

2.1 Proposed System

In my proposed system, study tools are available in a single place which will help students to save time and increase productivity. It provides an easy-to-use and secure environment with Centralized control by the administrator. It is a user-friendly app with amazing features like discussion forum, creating and sharing notes with others, finding books online, to-do, language translator, pdf reader.

My proposed system has several advantages

- ✓ User-friendly interface
- ✓ Fast access to the database
- ✓ Less error
- ✓ Search facility
- ✓ Easier Interaction and many more

2.2 System Scope

The project “Study Portal” has a wide scope, as It is not intended for a particular organization this project going to develop generic software, which can be used by any study organization. More-over, it provides facilities to its users and makes studying easier. Also the software is going to provide a huge amount of summary data

2.3 Feasibility Study

2.3.1 Technical Feasibility:

In this study we check whether developing the project is technically feasible or not

we will check the resources like hardware or software required to do this project is technically available or not

The tools and technology that were used in the making of the Study Portal.

- **Code Editor:** Visual Studio Code.
- **image source:** https://www.istockphoto.com
- **Database:** MySQL Database
- **Online Modules**: gTTS, googletrans, google book Api,
- **Programming Language:** Python (Django framework),

  HTML5, CSS, JavaScript, Bootstrap

• These mentioned above technologies are completely free for students. There were no other additional tools required to make this project.

• The simplicity of the project along with the facts mentioned above proves that this software is also technically feasible

2.3.2 Economic feasibility:

Economic feasibility is a term which is related to estimation of

cost required for the project

The resources that are required for this project are:

- Development machine: Any regular laptop/PC with a Quad Core CPU and Minimum Ram of 4G.
- Technical tools and software: As mentioned previously, the tools needed to develop this study portal are available to developers at no charge.

2.3.3 operational feasibility:

It checks if the software perform all the operations the way it should. It checks if the system is working in a perfect manner. It means the project must perform all the operational functions for which it is implemented. The project “STUDY PORTAL” through PYTHON(Django) TECHNOLOGY is operationally feasible. This project is well capable of providing a user good interface thus this project is able to work in effective and efficient manner.

Chapter 3: System Requirement Specifications

3.1 Software Requirement

Operating System: windows, mac, Linux

Language: Python (Django framework)

Database: MySQL

Code editor: Visual Studio Code

Browser: Google Chrome, Microsoft Edge, Firefox

3.2 Hardware Requirements

PC with 464 GB or more Hard disk.

PC with minimum 4 GB RAM.

3.3 Functional and non-functional requirement of the system:

- **Functional Requirements:** A functional requirements of the system describe what the system should do.

Functional requirement of my system are:-

User:

User must have to register with their basic credentials like unique username, unique email, first Name, last name branch, and password.

After having completed with the registration process the user have to login with their username and password thus creating their individual profile

User should able to join discussion forum to Post/Edit questions/answers

User should be able to create update delete to-do list.

User should be able to create, update, download and delete notes

User should be able to share, download notes and delete notes only which are uploaded by him.

User should be able to search notes by title and subject name.

user can use translator, pdf reader as well as search books online according to their needs.

Admin:

Admin have to login to admin panel to manage website.

admin should able to manage notes, users and posts, replies.

Notes should be published only if accepted by admin

- ## Non-Functional requirements:

Requirement that specifies criteria that can be used to judge the operation of a system are called non-functional requirements. Non-Functional requirement of my system are:

Performance:

The Study Portal is able to perform desired tasks in reasonable unit of time.

Reliability:

the system that I am developing can perform desired tasks as expected

The system does its work with more accuracy like

user registration to the system, user validation and authorization, creating, deleting

and updating into database.

Security:

The system will provide access to only legitimate users. It will be secure on network and only authorized person can use it.

Scalability:

The system will scalable to support extended number of users.

Maintainability:

The system will be easy to maintain and extend. Minor modification to the system would not cause harm to the running application

Chapter 4. System Design

4.1 Database Design

4.1.1 ER Diagram

The relation upon the system is structure through a conceptual ER Diagram. Which not only specifies the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.

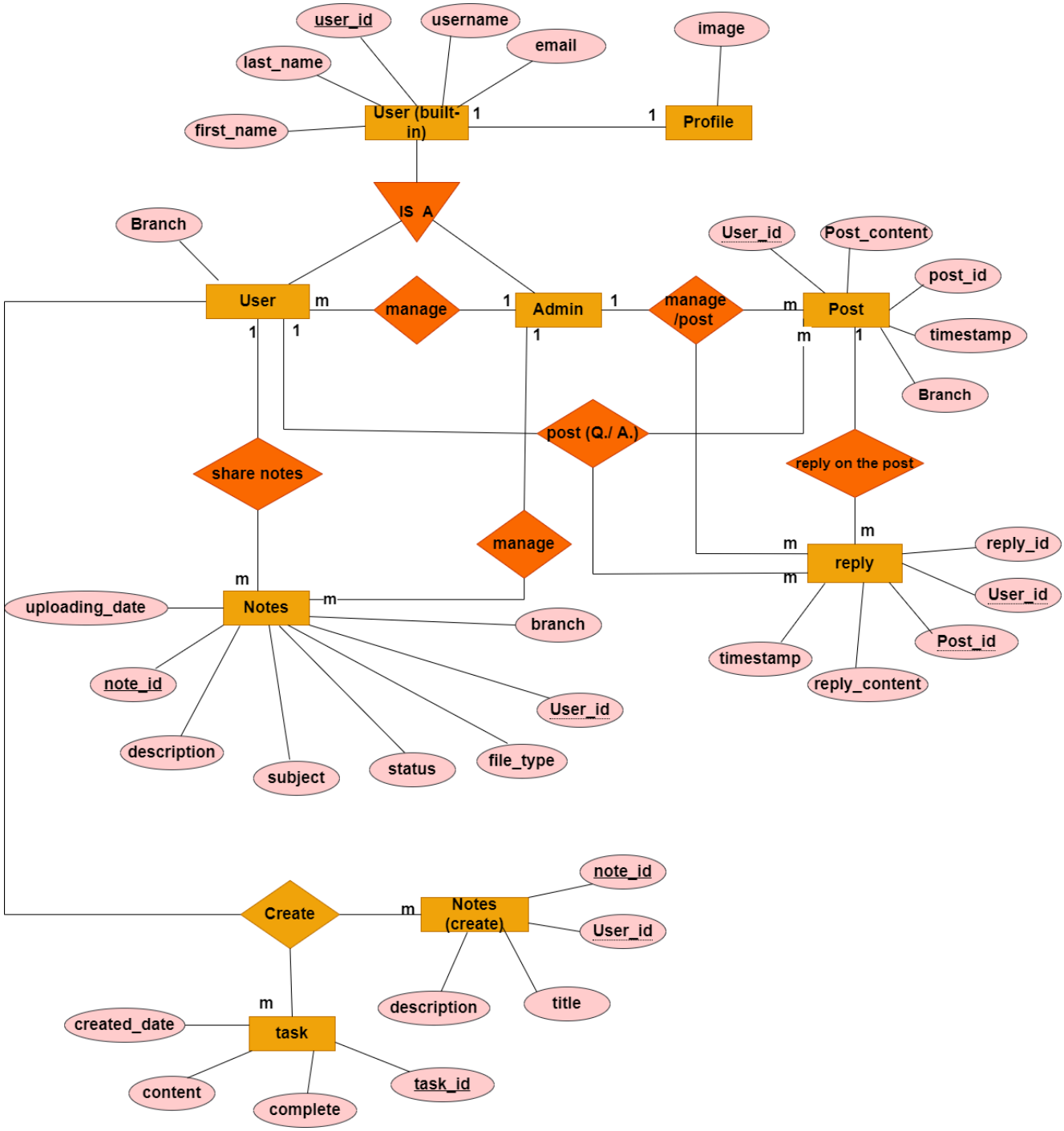
ER diagrams are used to sketch out the design of a database

This ER Diagram represents the model of Study Portal Entity.

The Entity Relationship Diagram of Study Portal shows all the

Visual instrument of database tables and the relation between User, Admin, Post, Reply, notes task, etc.

It used structure data to define the relationships between structured data and groups of study portal functionalities. The main entities of the system are User, Admin, Post, Reply, notes, books, task.


user_id
username
email
last_name
first_name
User (built-in)
1
1
Profile
image
IS A
Branch
User
m
manage
1
Admin
1
manage /post
m
Post
User_id
Post_content
post_id
timestamp
Branch
1
1
1
m
1
post (Q./ A.)
share notes
reply on the post
manage
m
m
reply
reply_id
User_id
Post_id
reply_content
timestamp
m
m
m
m
uploading_date
Notes
branch
note_id
User_id
description
subject
status
file_type
Create
m
Notes (create)
note_id
User_id
description
title
m
created_date
task
content
complete
task_id

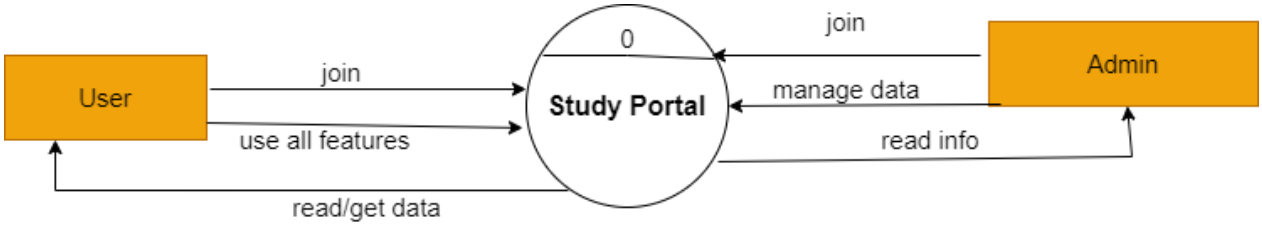
4.1.2 Data Flow Diagram

A Data Flow Diagram is the graphical representation of the flow of data from one component to another component of the system.

System Data Flow diagram often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. It normally consists of over-all application dataflow and processes of the system process. It contains all of the user flow and their entities such all the flow of system, User, Registration, login post reply note creating, notes sharing etc. All of the bellow diagrams have been used for the visualization of the data processing and structured design of the system process and working flow

Zero level DFD:

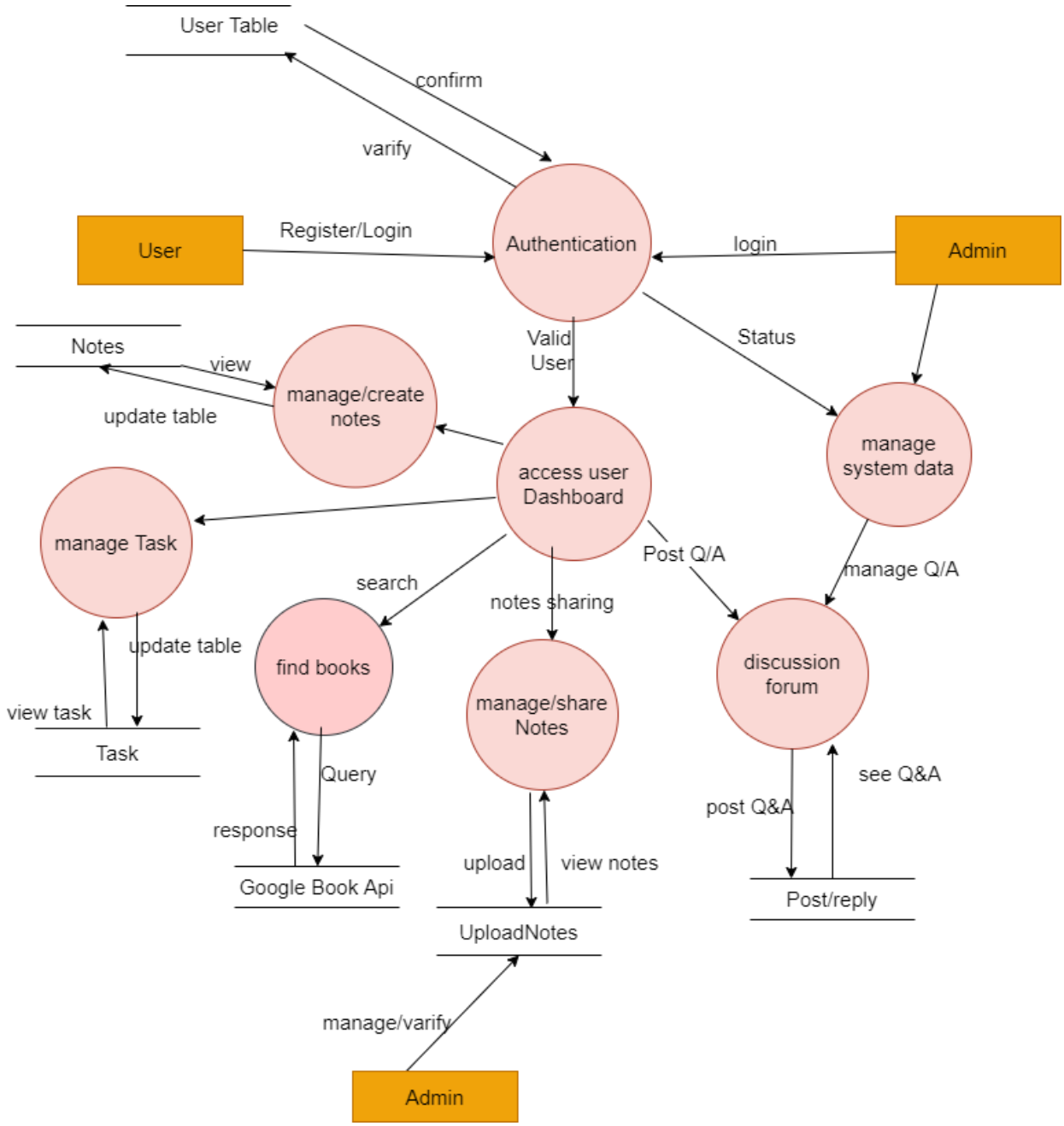
This is zero level DFD of study portal where I have elaborated the high-level process of the system. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view of the system, showing the system as a single high-level process, with its relationship to external entities of the study portal



First level DFD:

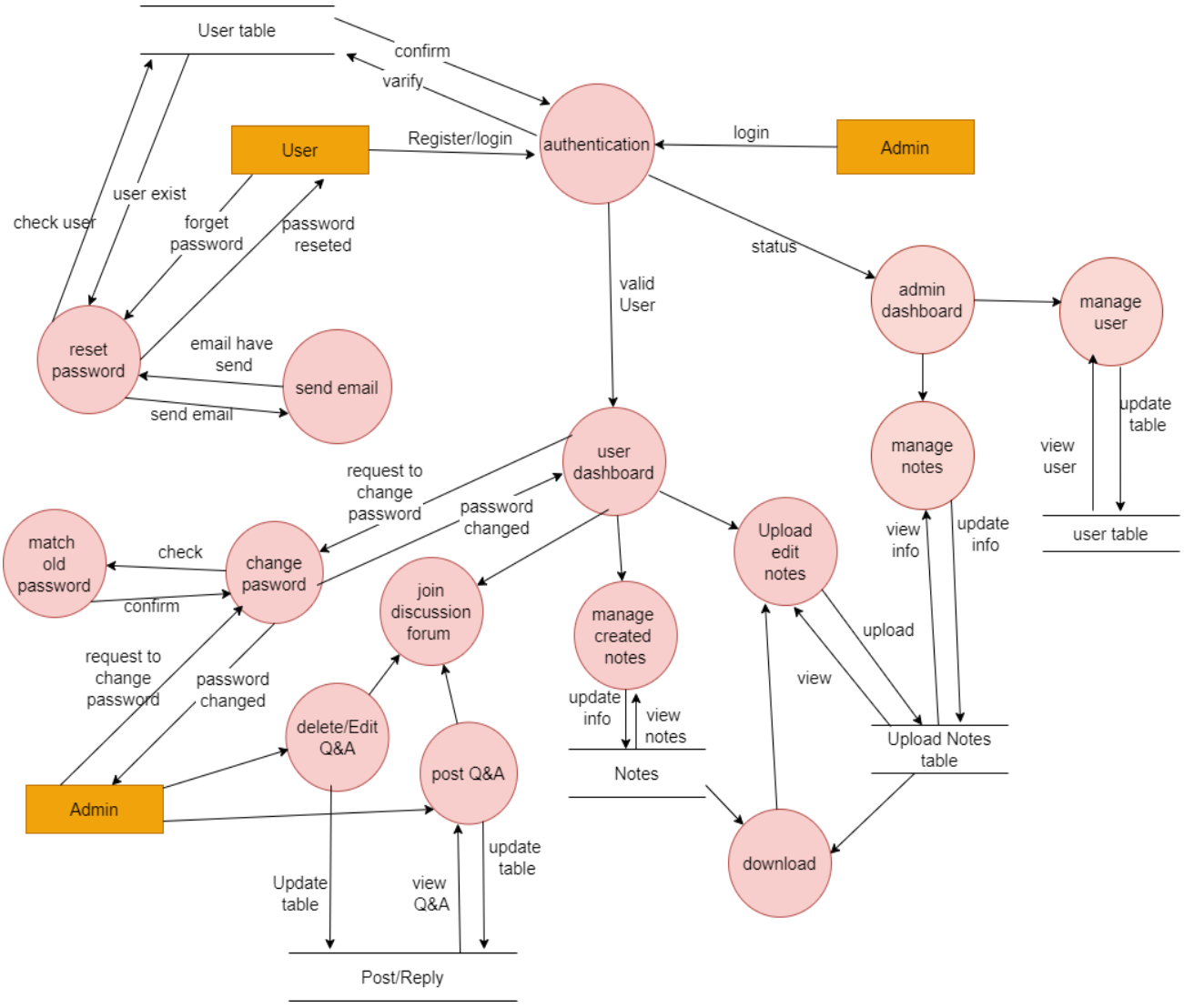
First level DFD of the study portal show how the system is divided into sub systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of

the functionality of the system as a whole.it also identifies internal data stores of the login registration note notes uploads post reply task that must be present in order for the system to do its job, and shows the flow of the data between various part of the system, user, admin, post reply notes, notes upload, task. DFD level 1 provides a more detailed breakout of pieces of the 1st level DFD. I highlight the main functionalities of the study portal



Second level DFD:

DFD level 2 then goes one step deeper into the level 1 of the study portal. It may require more functionalities of the system to reach the necessary level of detail about the study portal functioning. First level DFD shows how the system is divided into subsystems(process). The 2nd level DFD contains more details of authentication system, notes sharing, notes creating, post, reply, manage users



4.2 UML Diagrams

UML is linked with object-oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

Structural Diagrams

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable.

These static parts are represented by classes, interfaces, objects, components, and nodes.

Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

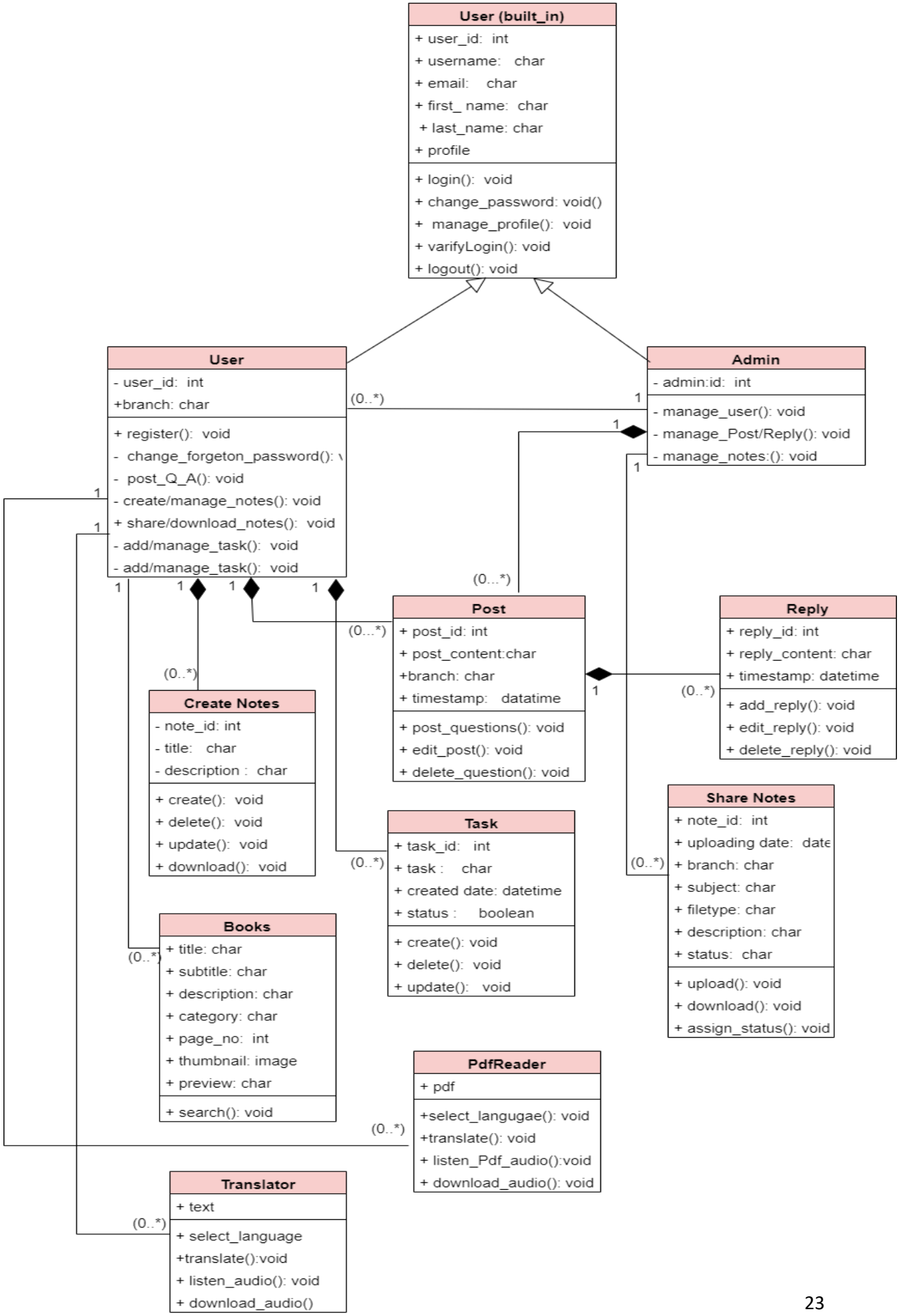
Behavioral Diagrams:

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered.

Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

4.2.1 Class Diagram:

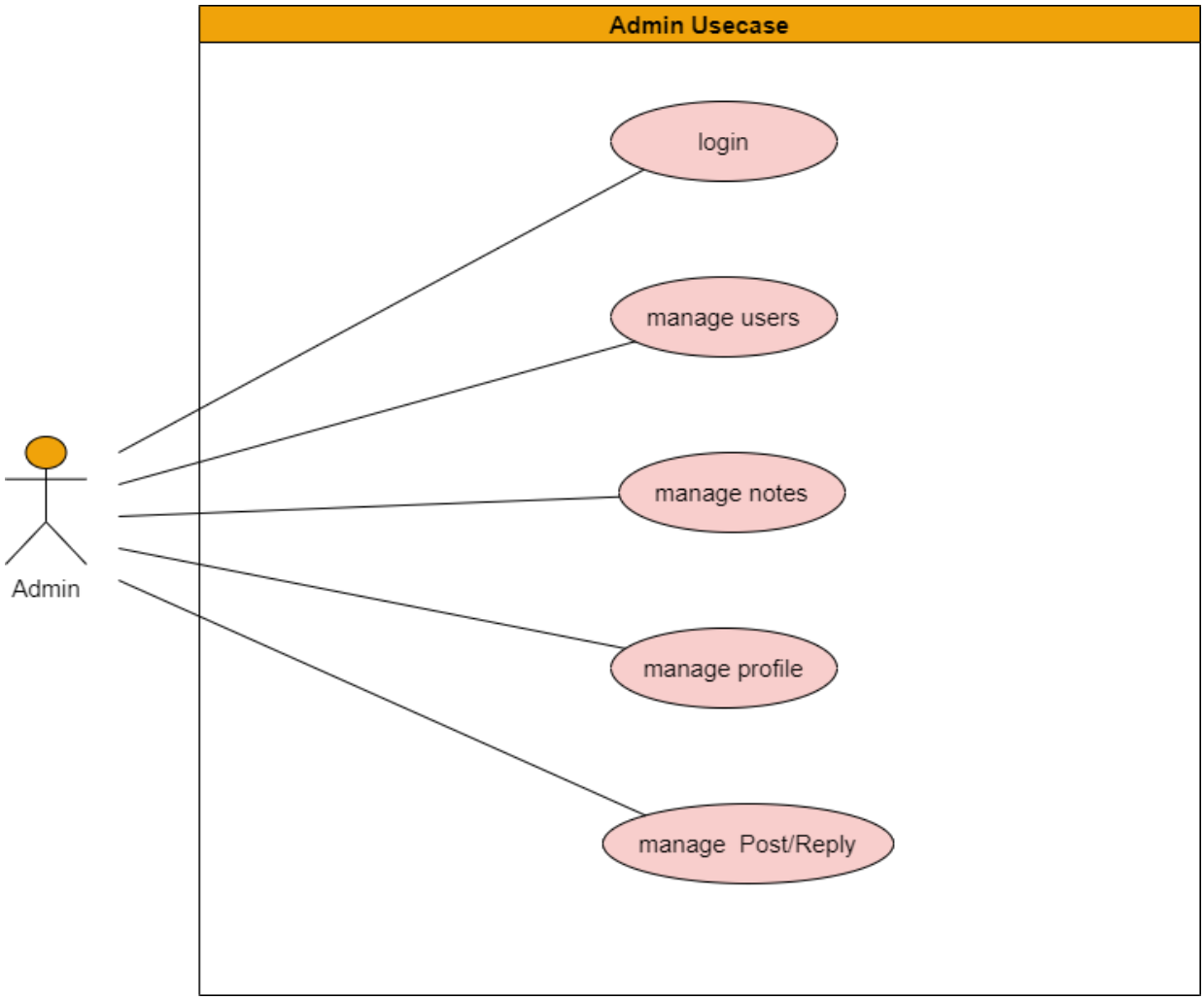
Class Diagram for Study Portal simply describes structure of study Portal class, attributes, methods or operations, relationship among objects. The main classes of the study portal are User, Admin, Post, Reply, Create Notes, Task, Share Notes, Books, Pdf-Reader, Translator.


User (built_in)
+ user_id: int
+ username: char
+ email: char
+ first_ name: char
+ last_name: char
+ profile
+ login(): void
+ change_password: void()
+ manage_profile(): void
+ varifyLogin(): void
+ logout(): void
User
- user_id: int
+branch: char
+ register(): void
- change_forgeton_password():
- post_Q_A(): void
- create/manage_notes(): void
+ share/download_notes(): void
- add/manage_task(): void
- add/manage_task(): void
Admin
- admin:id: int
- manage_user(): void
- manage_Post/Reply(): void
- manage_notes:(): void
Post
+ post_id: int
+ post_content:char
+branch: char
+ timestamp: datatime
+ post_questions(): void
+ edit_post(): void
+ delete_question(): void
Reply
+ reply_id: int
+ reply_content: char
+ timestamp: datetime
+ add_reply(): void
+ edit_reply(): void
+ delete_reply(): void
Create Notes
- note_id: int
- title: char
- description : char
+ create(): void
+ delete(): void
+ update(): void
+ download(): void
Task
+ task_id: int
+ task : char
+ created date: datetime
+ status : boolean
+ create(): void
+ delete(): void
+ update(): void
Share Notes
+ note_id: int
+ uploading date: date
+ branch: char
+ subject: char
+ filetype: char
+ description: char
+ status: char
+ upload(): void
+ download(): void
+ assign_status(): void
Books
+ title: char
+ subtitle: char
+ description: char
+ category: char
+ page_no: int
+ thumbnail: image
+ preview: char
+ search(): void
PdfReader
+ pdf
+select_langugae(): void
+translate(): void
+ listen_Pdf_audio():void
+ download_audio(): void
Translator
+ text
+ select_language
+translate():void
+ listen_audio(): void
+ download_audio()
(0..*)
1

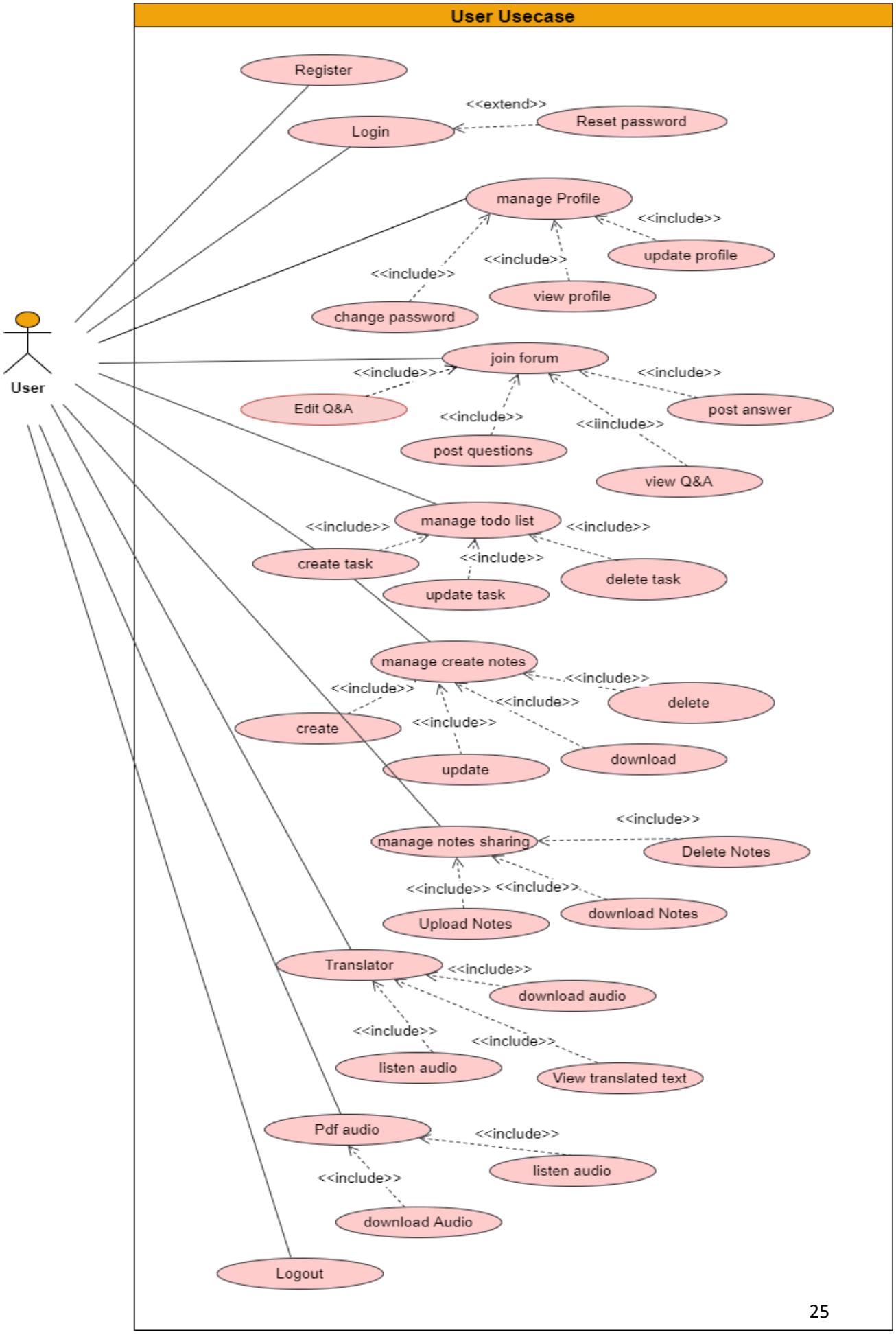
4.2.2 Use Case Diagram

These Use Case Diagrams are graphical depiction of the interactions among the elements of the Study Portal. It represents the methodology used in the system analysis to identify, clarify, and organize system requirements of Study Portal. The main actors of Study Portal in the following Use Case diagrams are Admin and system User. who performs the different type of Use Cases such as manage system data, manage users, manage posts/replies manage notes, create notes, manage task etc .

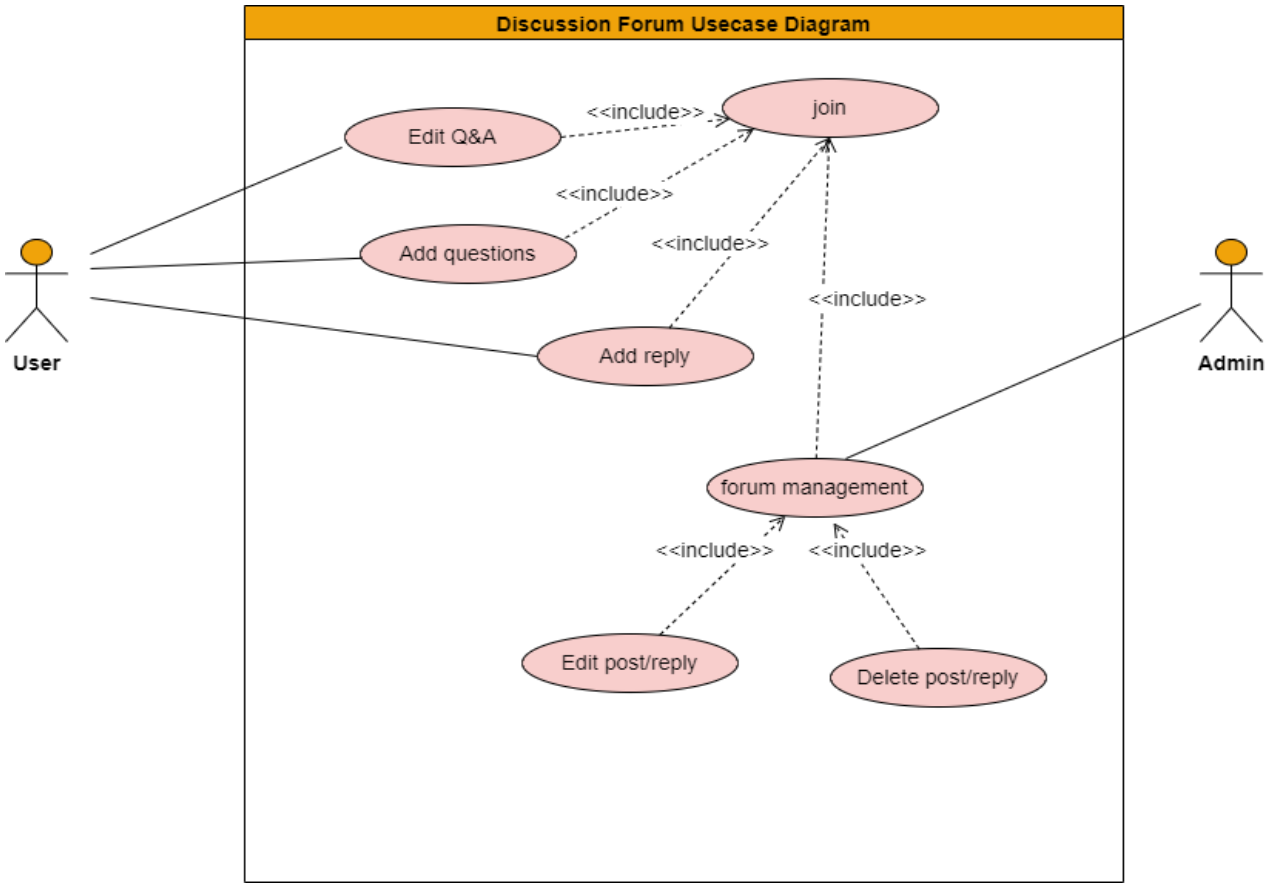
Admin Use Case Diagram:



User Use Case Diagram:



Discussion Forum Use Case Diagram:

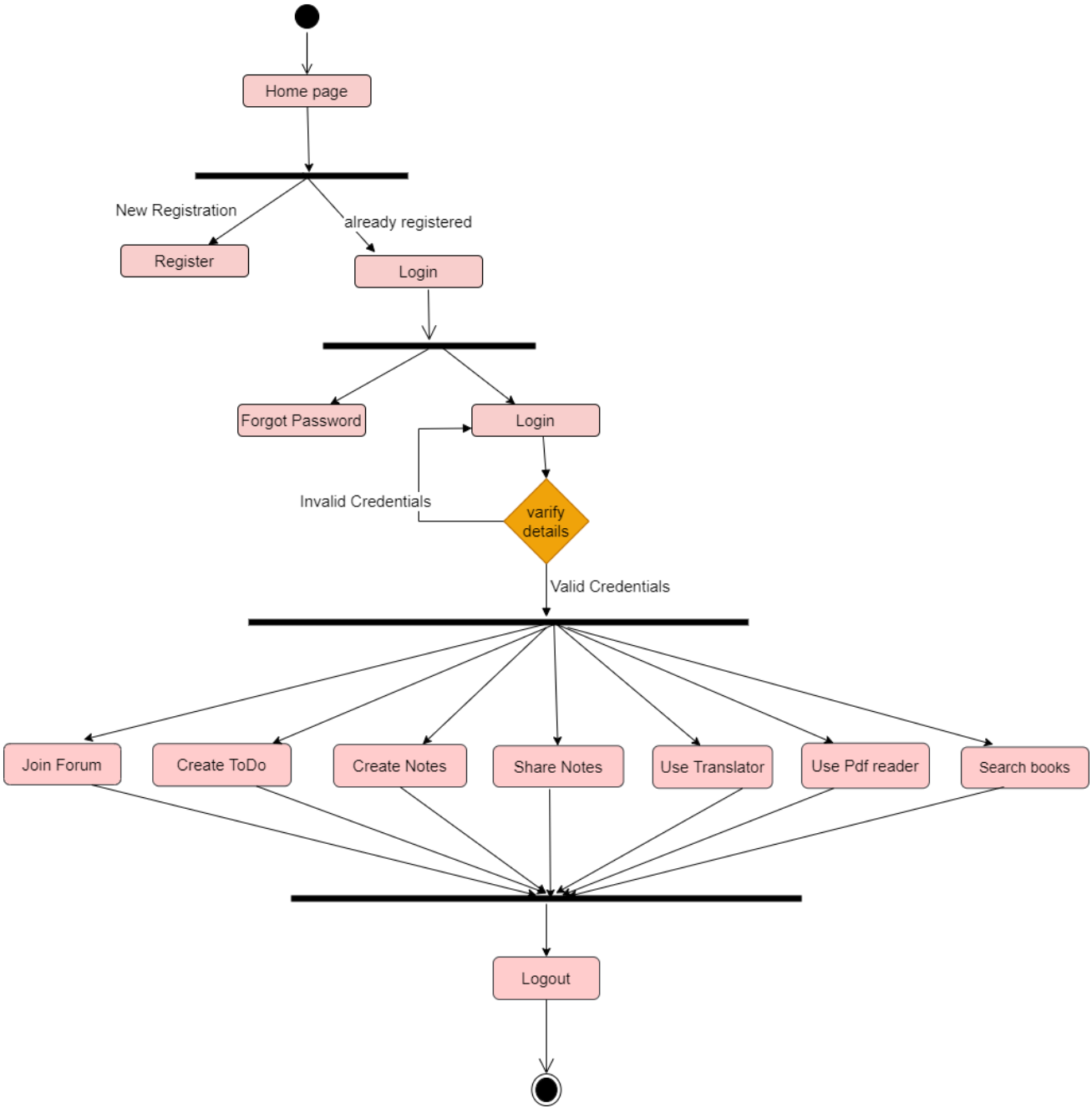


4.2.3 Activity Diagram

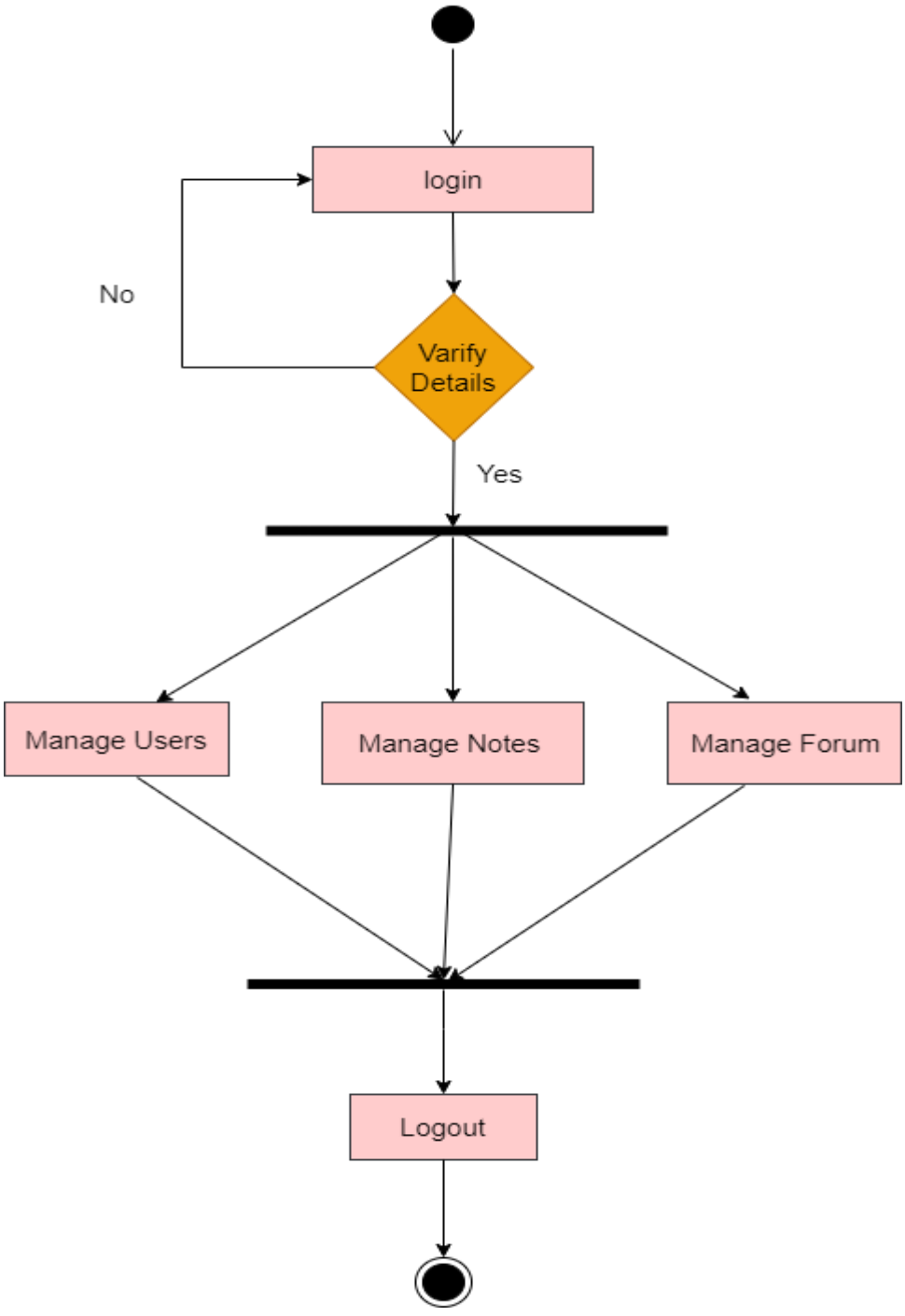
Activity UML diagram of Study portal show the flows between the activity of the Users, Admin, discussion forum, notes sharing, notes creating, task management etc. Below there are two activity diagrams.

1. **User activity:** In the User activity diagram shows the flow of activities perform by User
2. **Admin activity:** In the Admin activity diagram shows the flow of activities perform by Admin

User Activity Diagram:



Admin Activity Diagram:

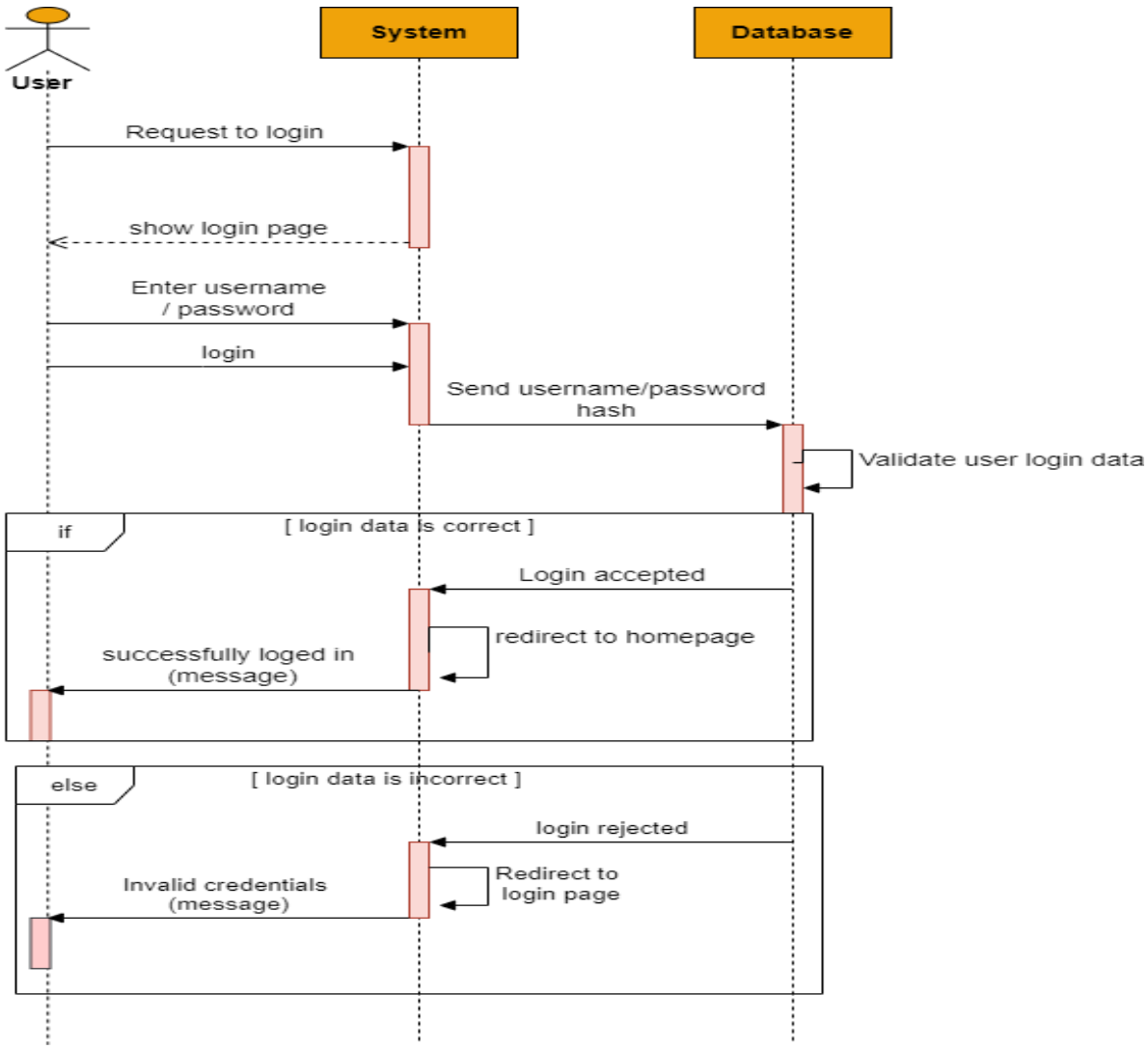


4.2.4 Sequence Diagram

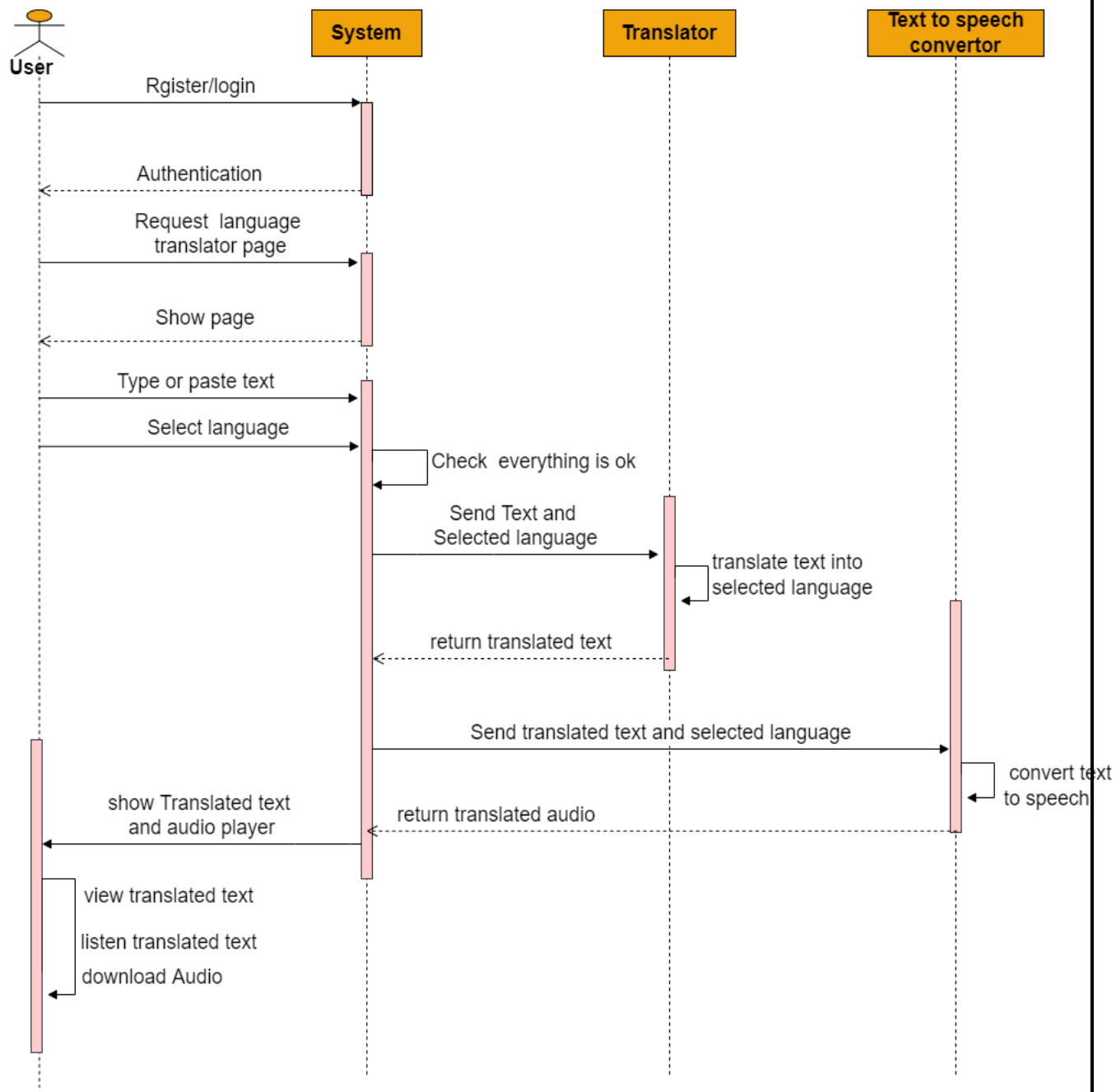
The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios.

It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time.

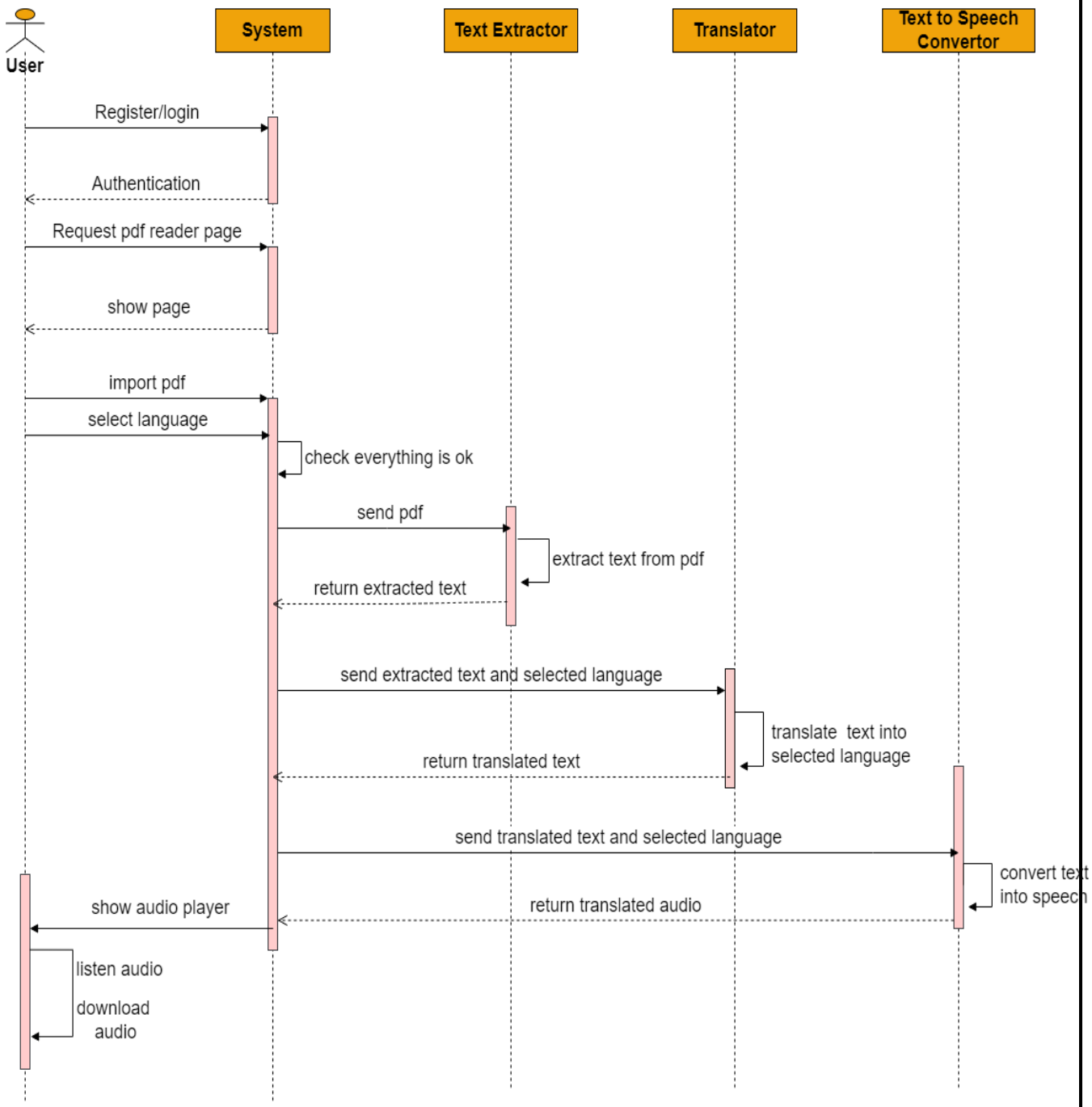
User Login Sequence



Language Translator Sequence Diagram:



Pdf Reader Sequence Diagram:



4.3 Data Dictionary

User Registration:

Field	Type	Key	Null	Default
id	int(11)	PRI	No	None
username	varchar(150)		No	None
first_name	varchar(150)		No	None
last_name	varchar(150)		No	None
email	varchar(254)		No	None
password	varchar(128)		No	None
last_login	datetime(6)		Yes	Null
date_joined	datetime(6)		No	None
branch	Varchar(100)		No	None

User Profile:

Field	Type	Key	Null	Default
id	bigint(20)	PRI	No	None
image	varchar(100)		No	None
user_id	int(11)	MUL	No	None

Discussion Post:

Field	Type	Key	Null	Default
Id	bigint(20)	PRI	No	None
post_content	longtext		No	None
timestamp	datetime(6)		No	None
branch	varchar(100)		No	None
user_id	Varchar(100)	MUL	No	None

Discussion Reply:

Field	Type	Key	Null	Default
id	bigint(20)	PRI	No	None
reply_content	longtext		No	None
timestamp	datetime(6)		No	None
image	Varchar(100)		No	None
post_id	bigint(20)	MUL	No	None
user_id	Int(11)	MUL	No	None

Create Notes:

Field	Type	Key	Null	Default
id	bigint(20)	PRI	No	None
title	Varchar(200)		No	None
description	longtext		No	None
user_id	Int(11)	MUL	No	None

ToDo:

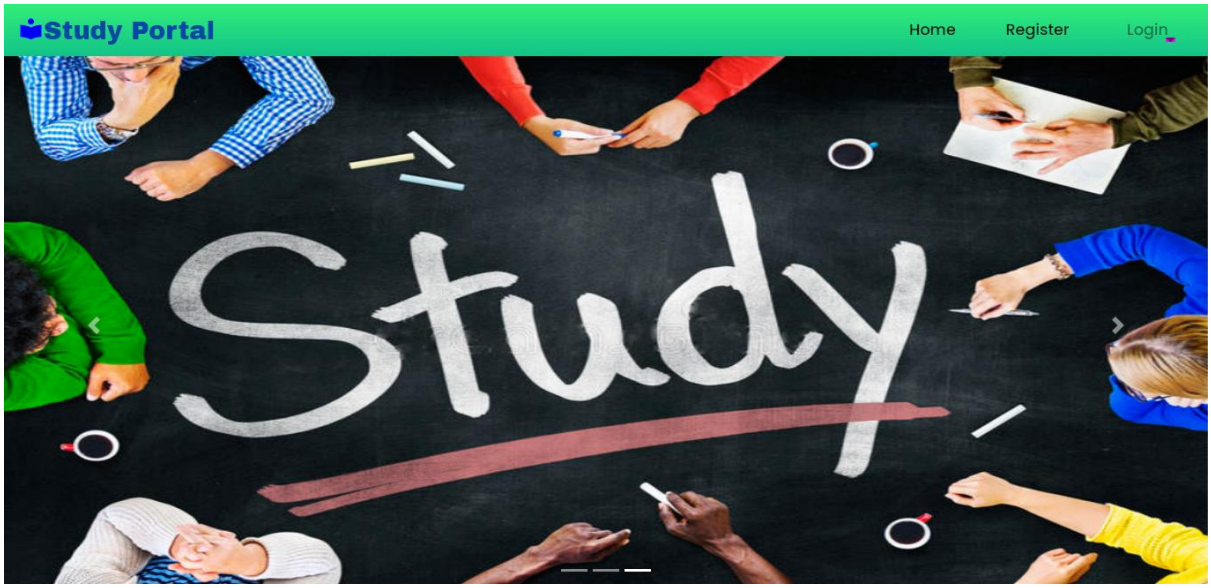
Field	Type	Key	Null	Default
id	bigint(20)	PRI	NO	None
content	varchar(50)		NO	None
date_created	datetime(6)		NO	None
complete	tinyint(1)		NO	None
user_id	int(11)	MUL	NO	None

Notes Sharing:

Field	Type	Key	Null	Default
id	bigint(20)	PRI	No	None
uploading_date	varchar(30)		No	None
branch	varchar(30)		No	None
subject	varchar(50)		No	None
notesfile	varchar(100)		No	None
filetype	varchar(30)		No	None
description	varchar(200)		Yes	Null
status	varchar(15)		No	None
user_id	int(11)	MUL	No	None

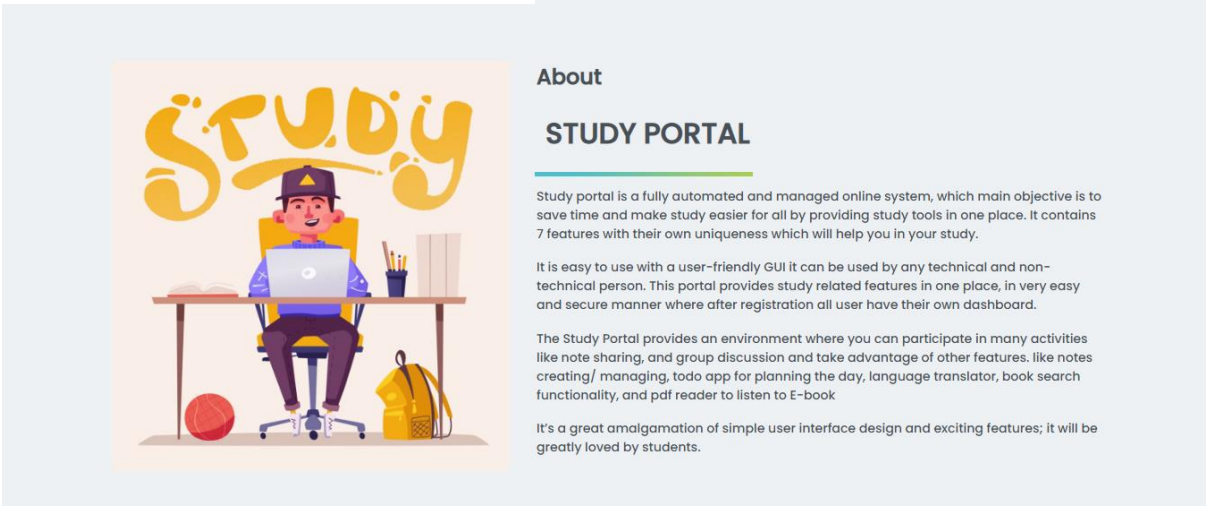
Chapter 5: Screenshots

1) Study Portal Home page



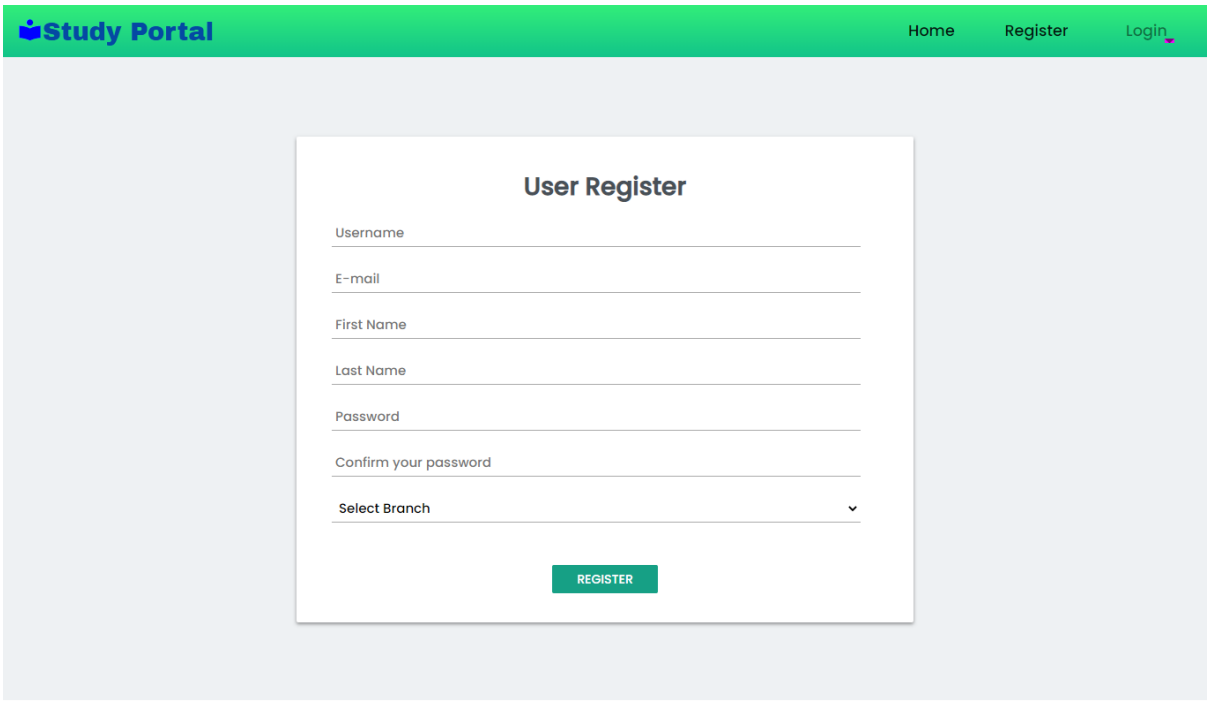
2)

3) About

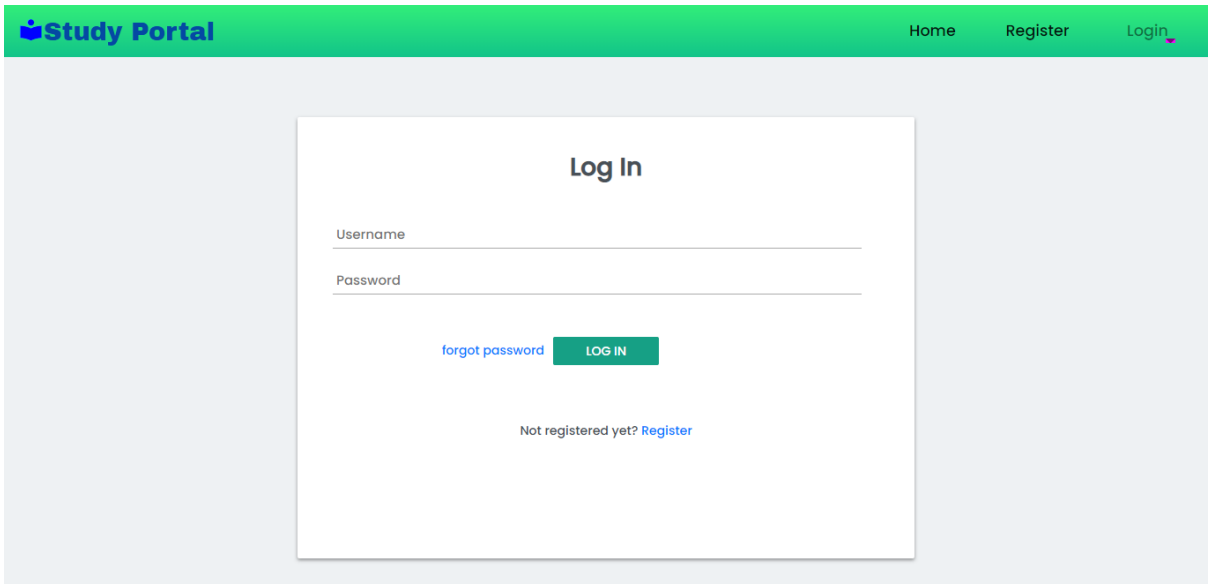


User Section Screenshots:

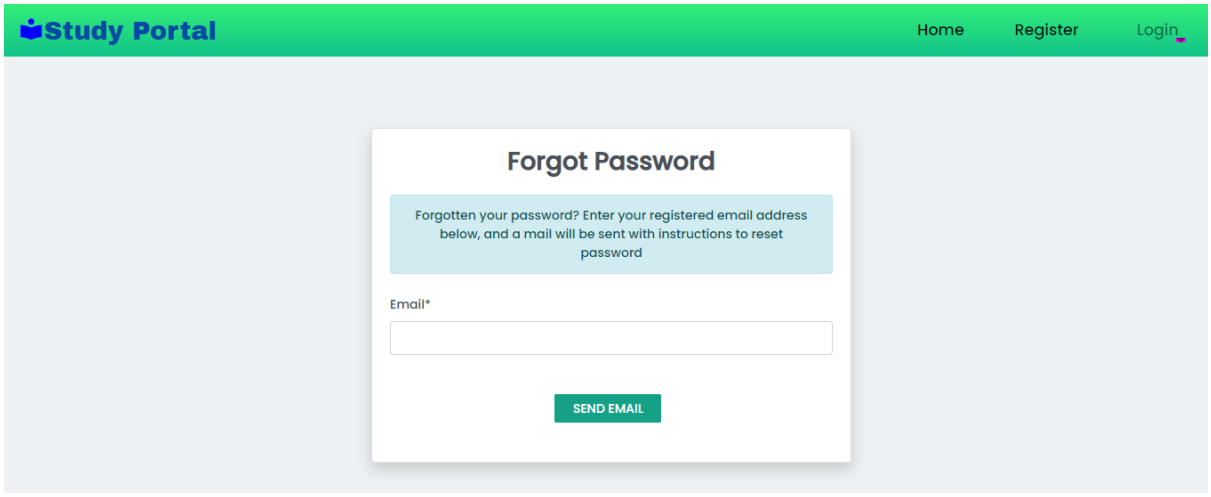
4) User Registration:



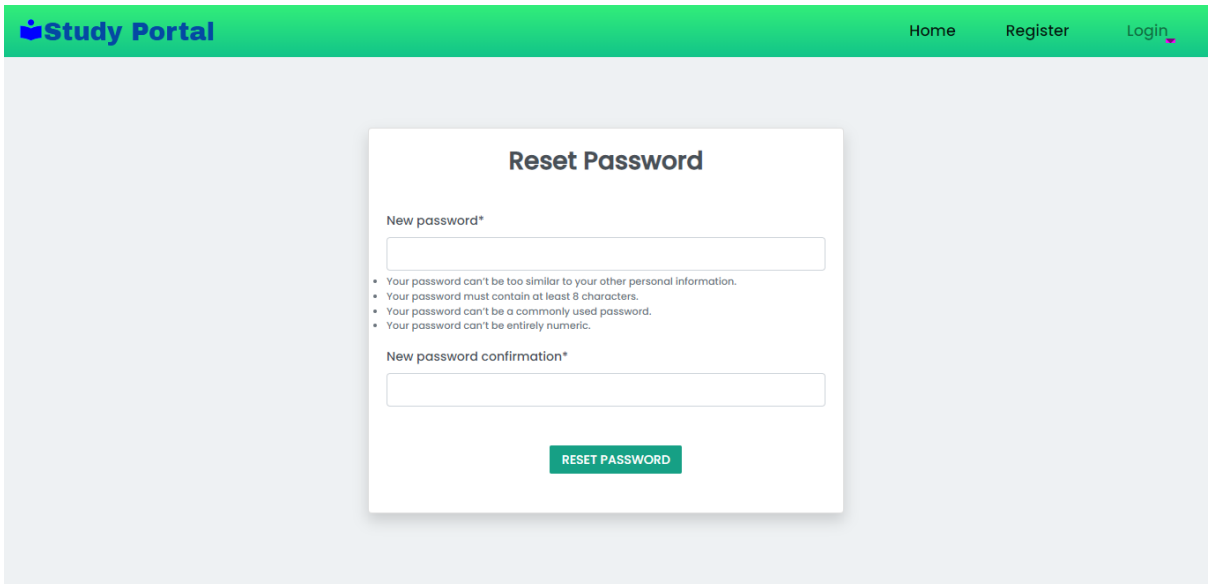
5) User Login:



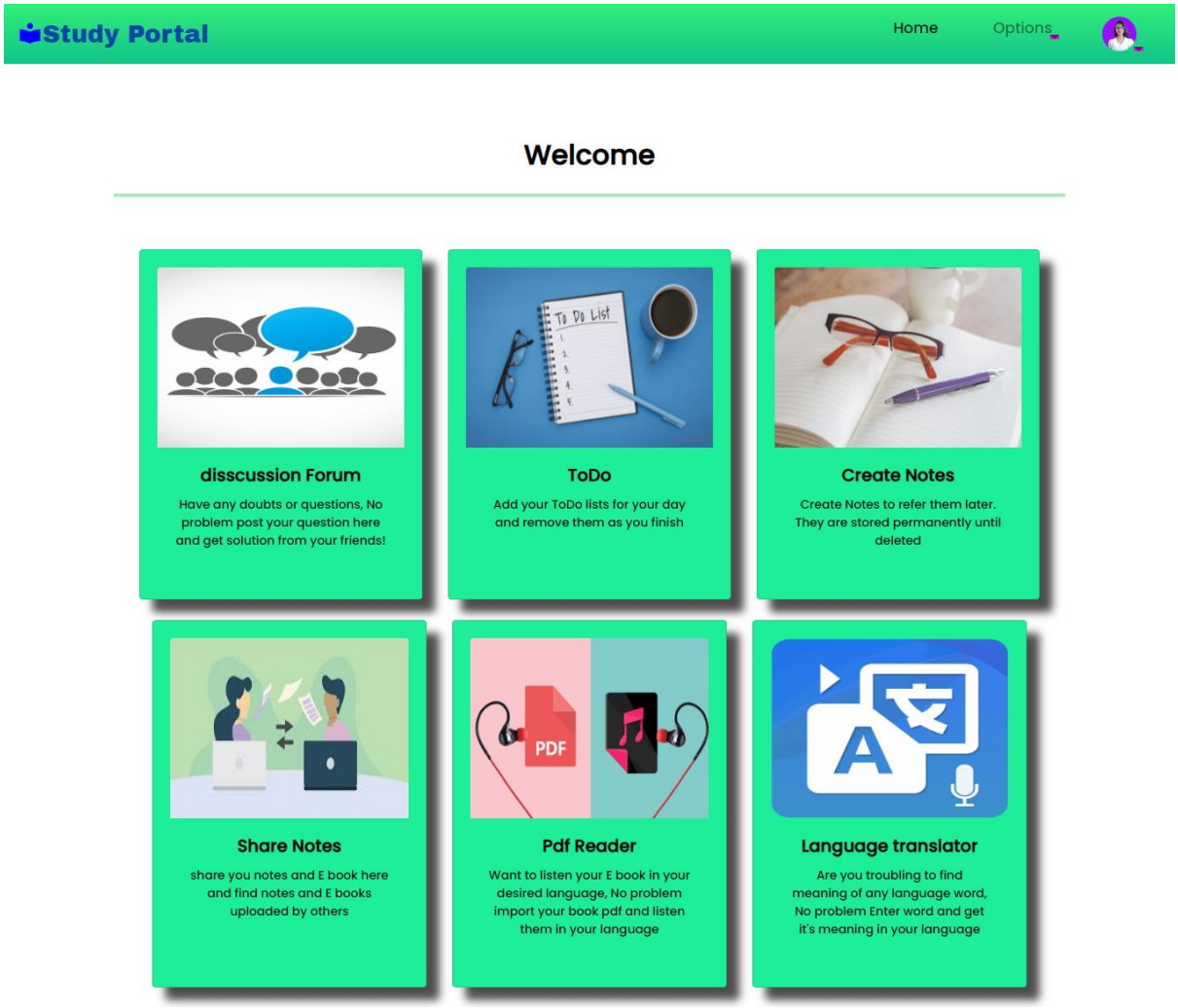
6) Forget Password:



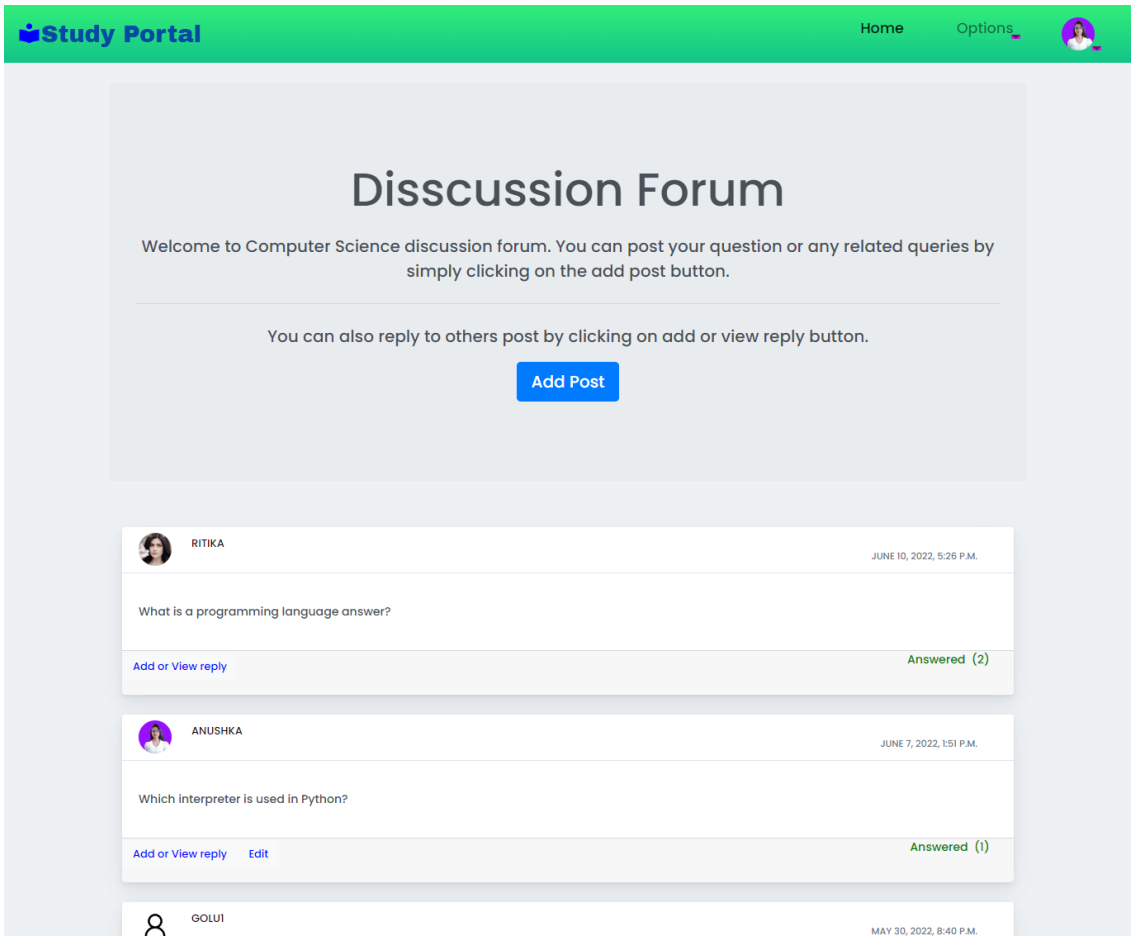
7) Reset Password:



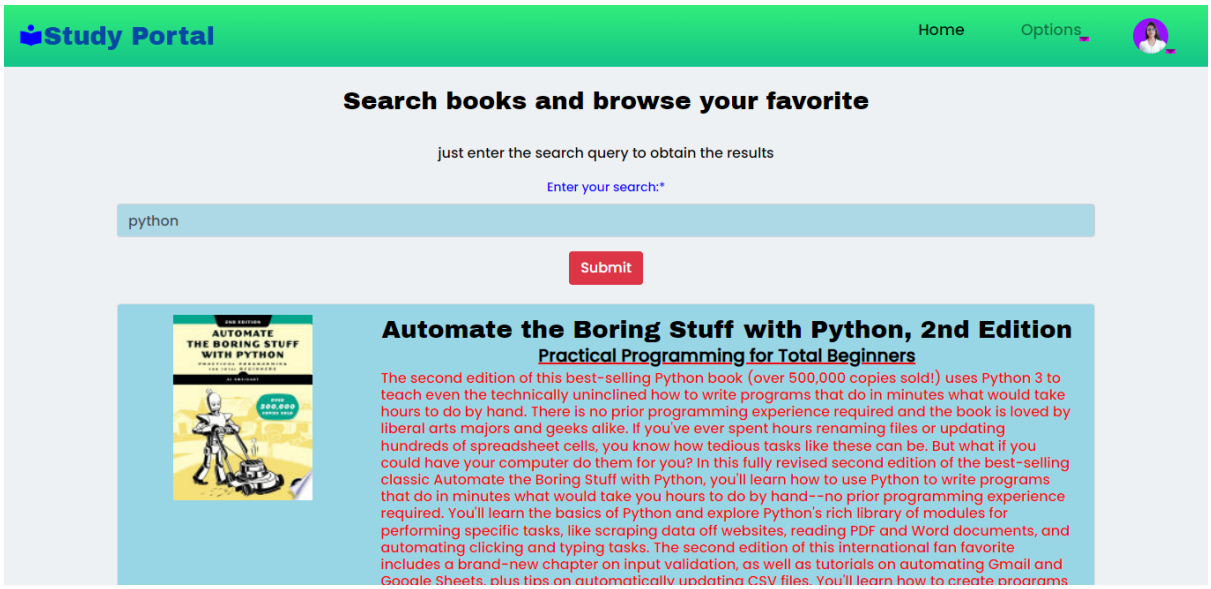
8) User Home page:



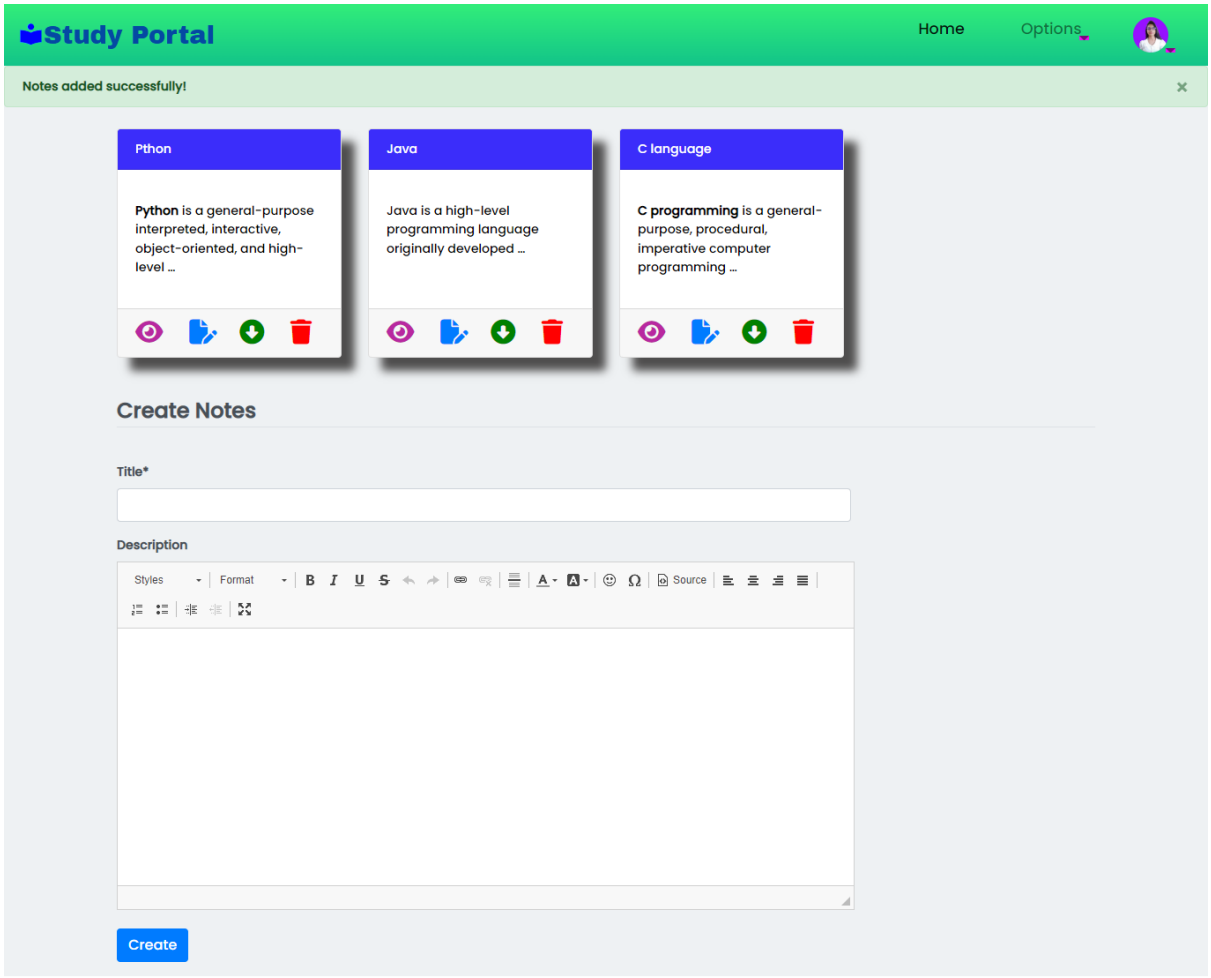
9) Forum:



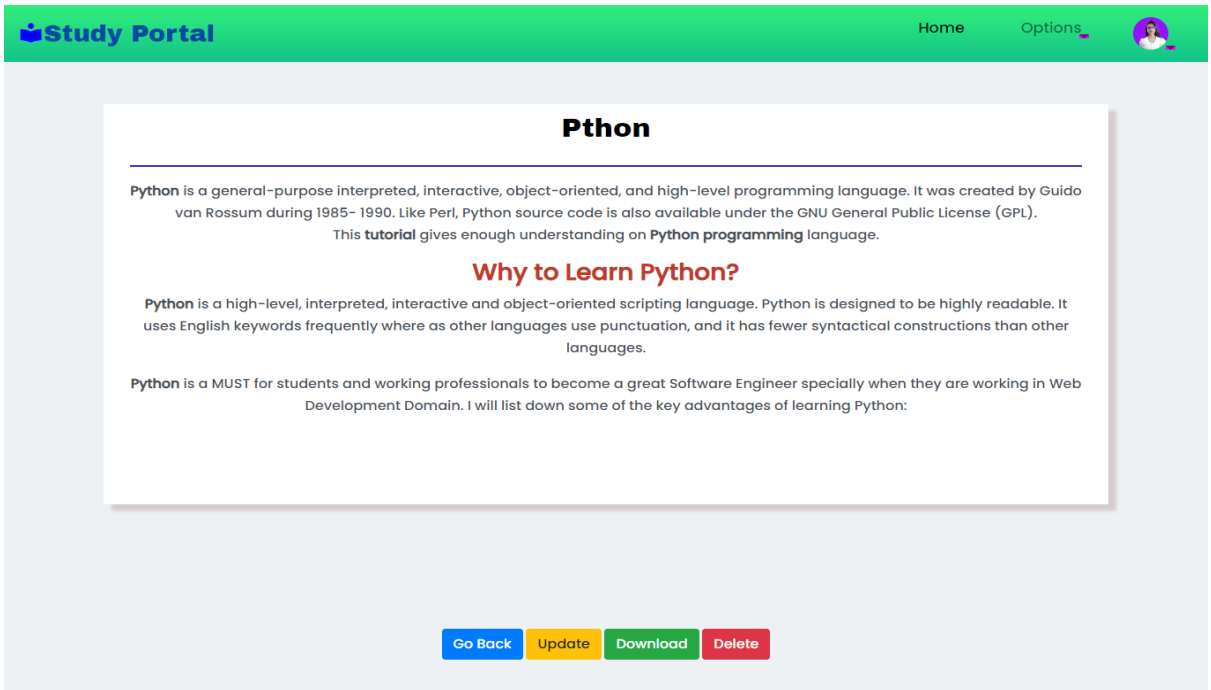
10) Book search:



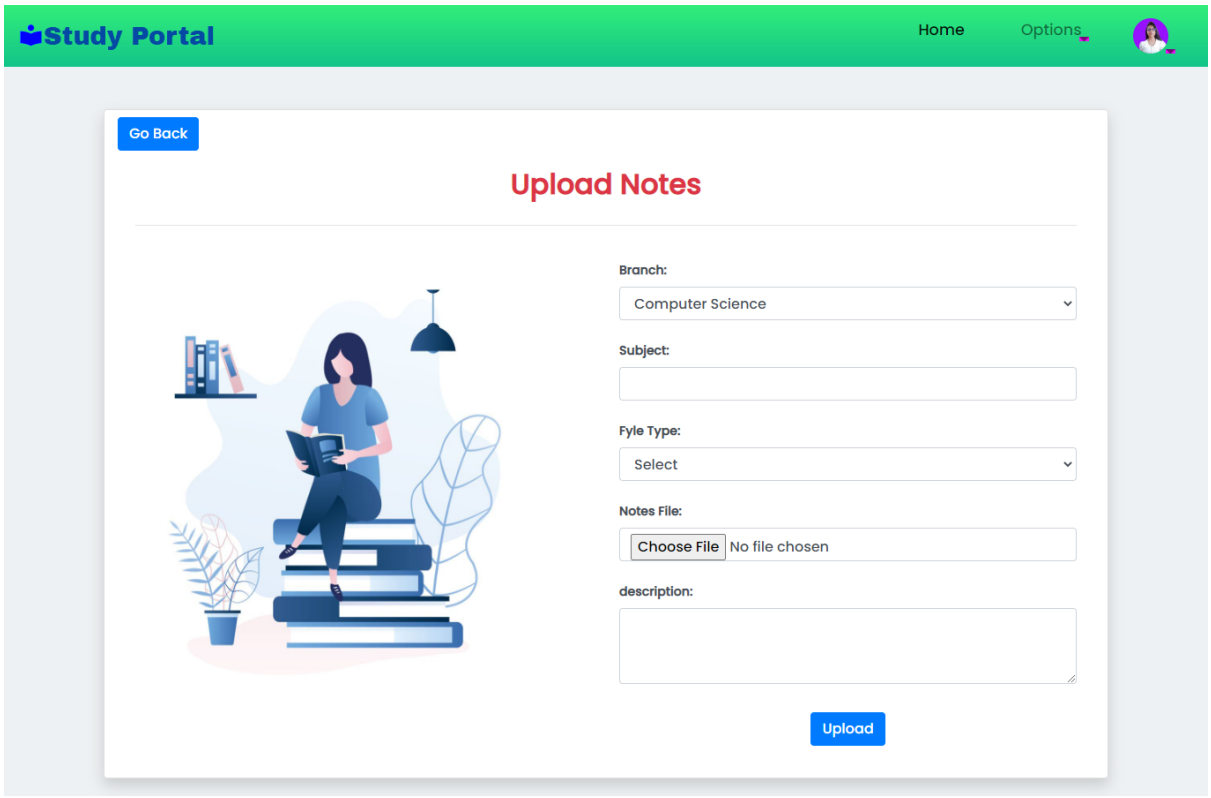
11) Create Notes:



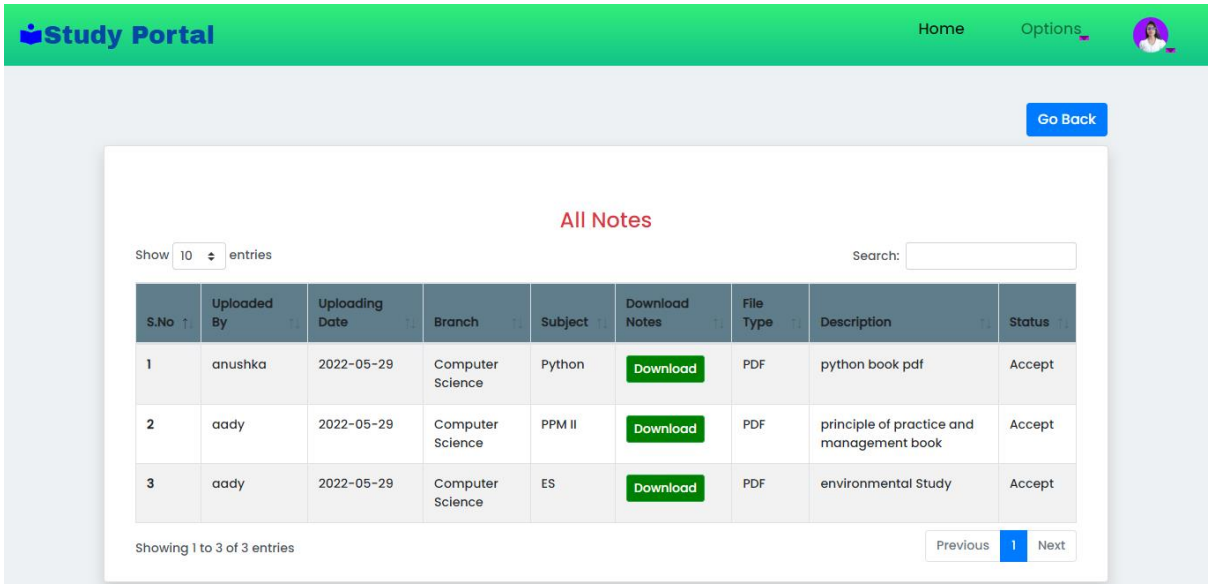
12) Created Notes details:



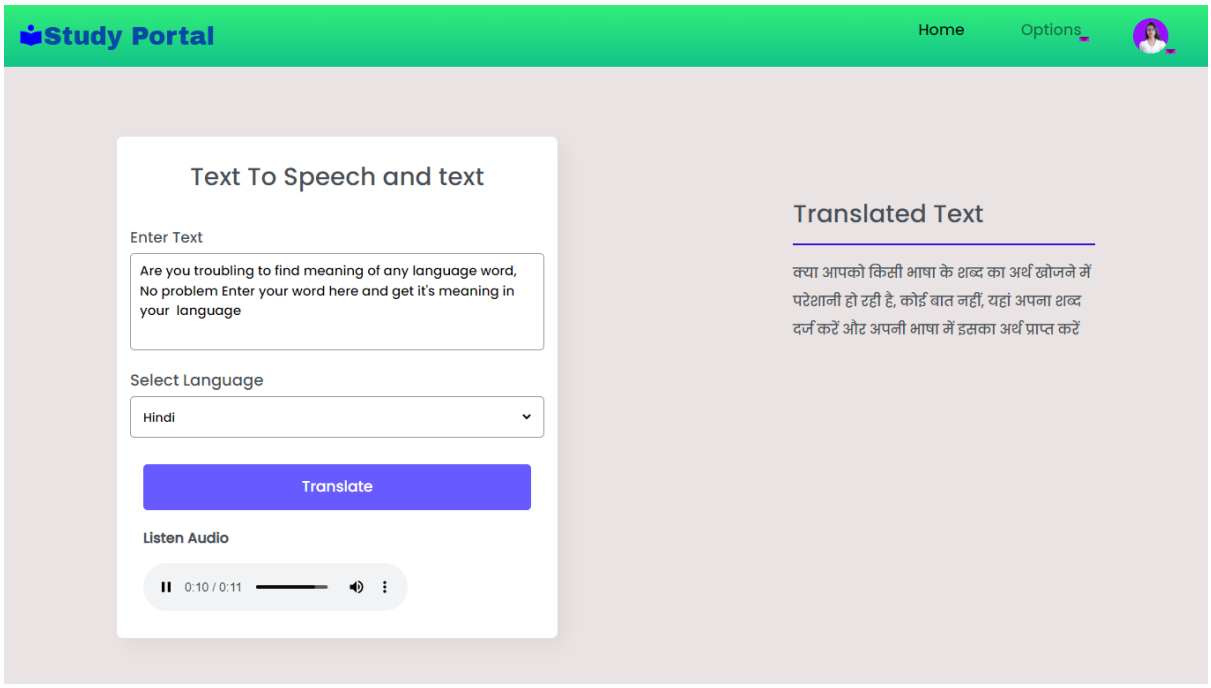
13) Upload Notes:



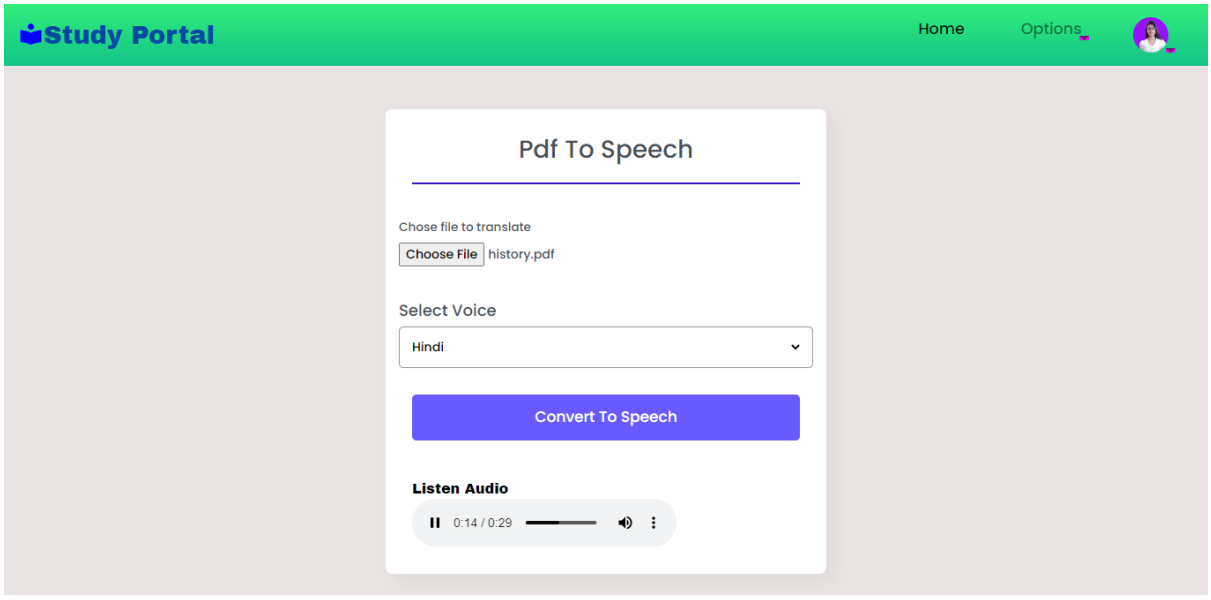
14) All Notes:



15) Language Translator:

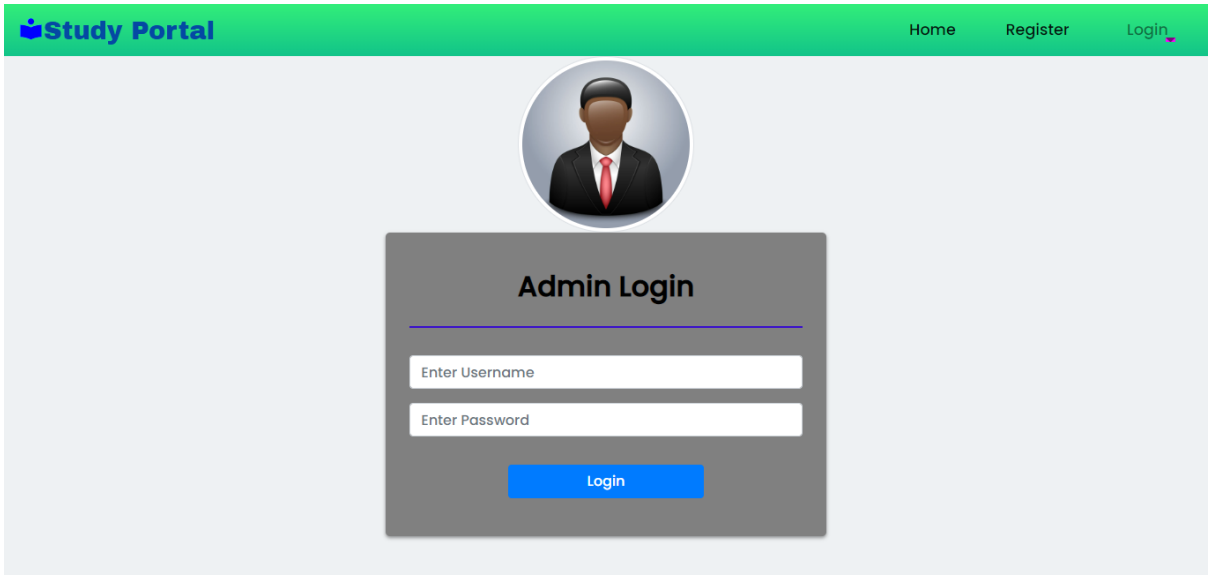


16) Pdf Reader:

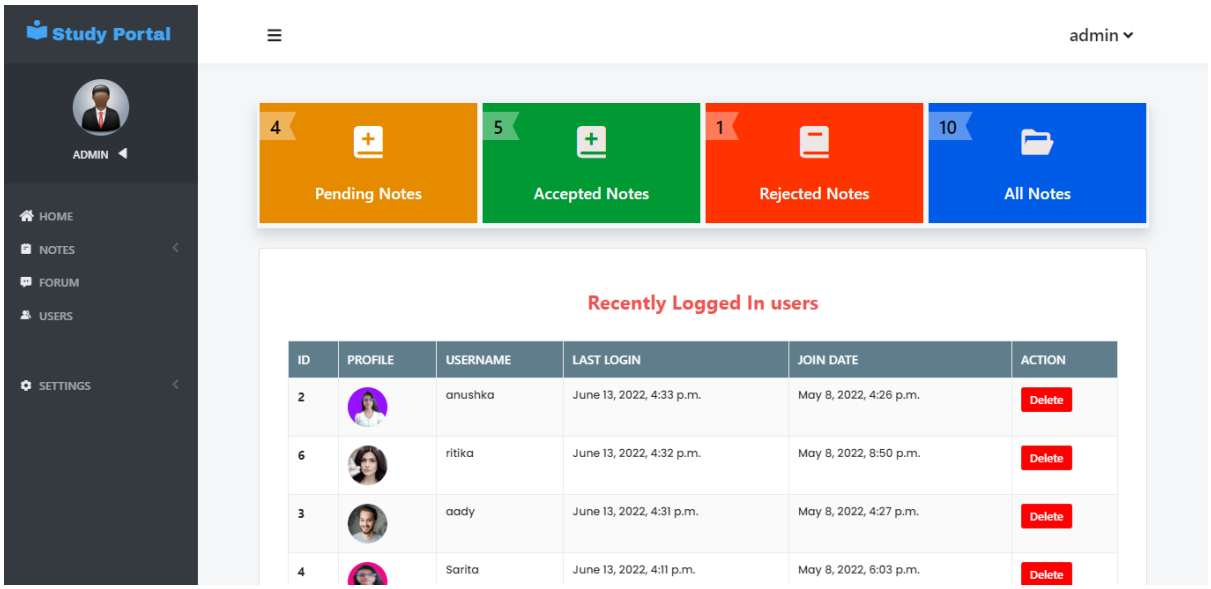


Admin Section Screenshots:

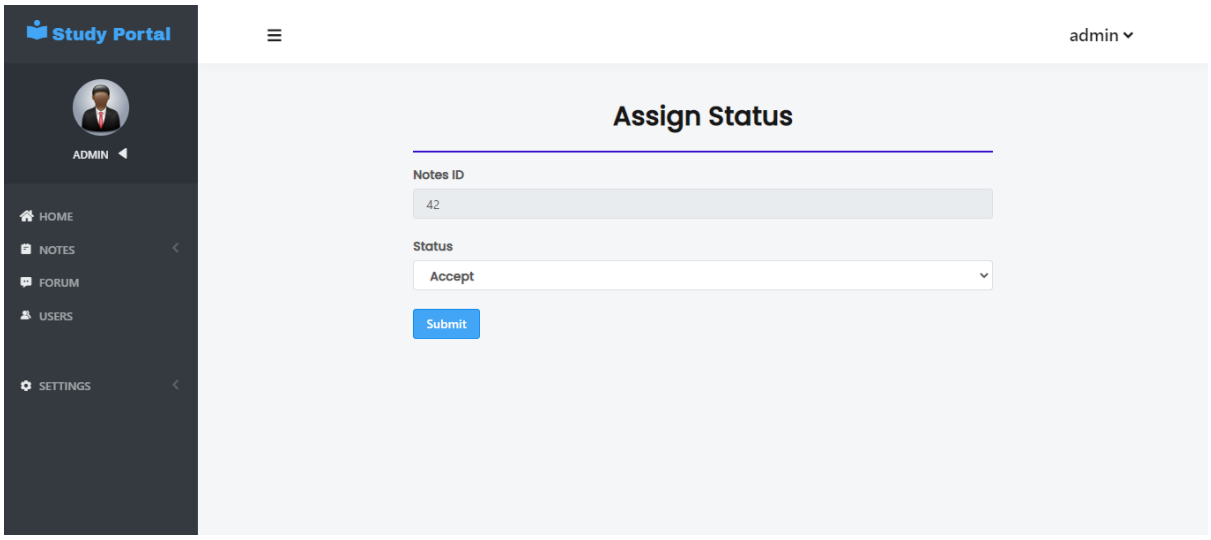
17) Admin Login:



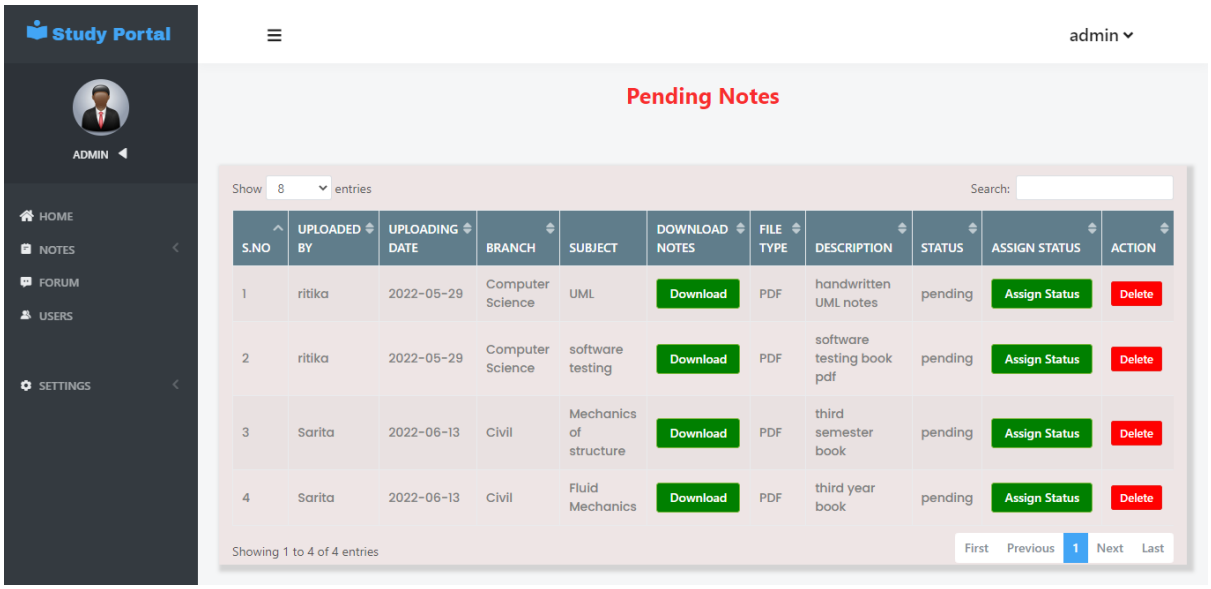
18) Admin Home page:



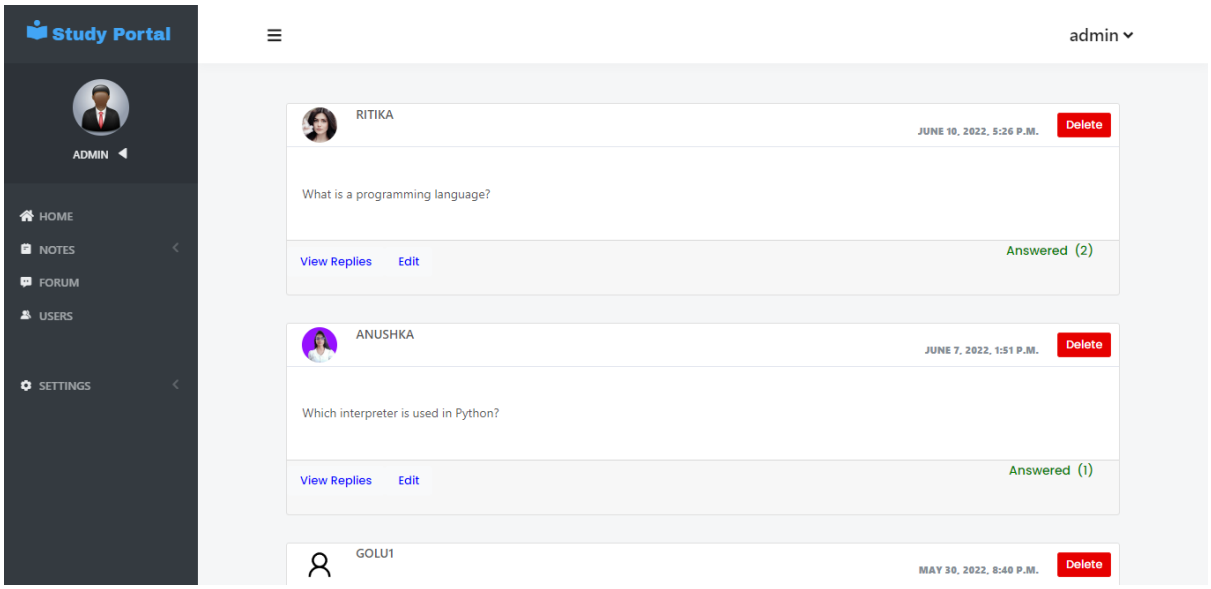
19) Assign Status to Uploaded Notes:



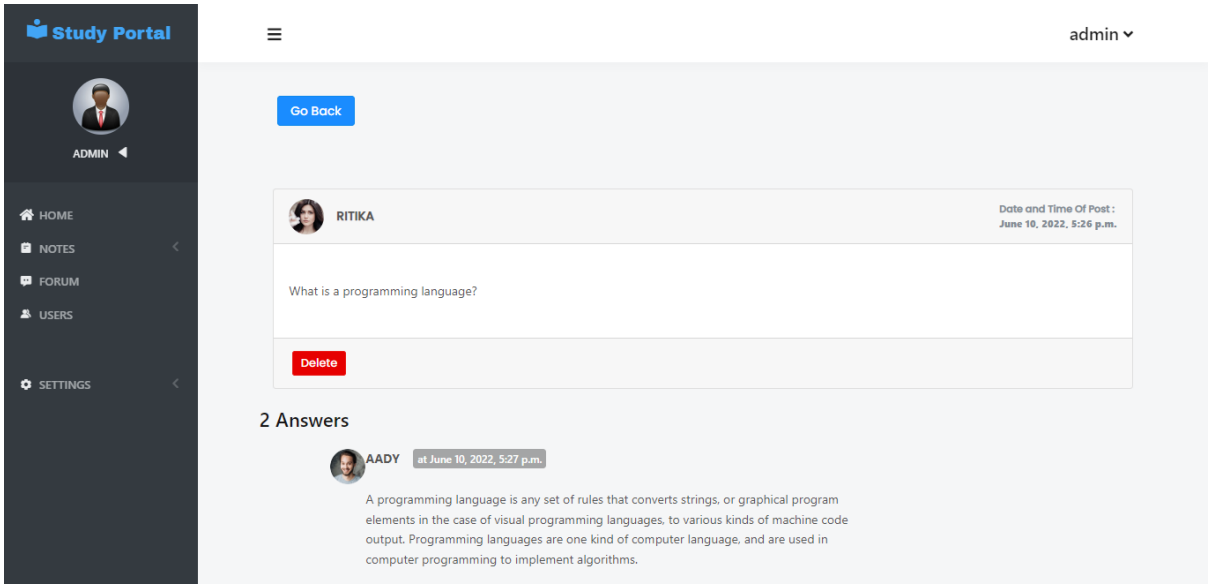
20) Notes Upload Request page:



21) Manage Forum:



22) Manage replies:



Chapter 6: Implementation Details

6.1 Implementation Tool and Technology

python:

Python is a high-level, interpreted, general-purpose programming language. It is often used to build websites and software, automate tasks, and conduct data analysis.

Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today.

A survey conducted by industry analyst firm Red Monk found that it was the second-most popular programming language among developers in 2022**.**

Python Web Framework:

A framework is a collection of modules or packages which helps in writing web applications. While working on frameworks in python we don't have to worry about the low-level details such as protocols, sockets or thread management. In this project I have used Django framework.

Advantages Of Frameworks

- Open-source
- Good documentation
- Efficient
- Secure
- Integration

Why Use A Framework?

Frameworks make it easier to reuse the code for common HTTP operations.

They structure the projects in a way so that the other developers with the knowledge of the framework can easily maintain and build the application.

Django:

Django is a free and open-source full-stack python framework, it includes all the necessary features by default.

that enables rapid development of secure and maintainable websites. Built by experienced developers,

Django takes care of much of the hassle of web development, so we can focus on writing our app without needing to reinvent the wheel.

It is free and open source, has a thriving and active community, great documentation, and many options for free.

Visual studio code:

One of the coolest code editors available to programmers, Visual Studio Code, is an open-source, extensible, light-weight editor available on all platforms.

It's these qualities that make Visual Studio Code from Microsoft very popular, and a great platform for Python development.

MSQL server:

The MySQL server provides a database management system with querying and connectivity capabilities,

as well as the ability to have excellent data structure and integration with many different platforms.

It can handle large databases reliably and quickly in high-demanding production environments.

The MySQL server also provides rich function such as its connectivity, speed, and security that make it suitable for accessing databases.

The MySQL server works in a client and server system. This system includes a multiple-threaded SQL server that supports varied backends,

different client programs and libraries, administrative tools, and many application programming interfaces (API)s.

MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL

is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the
- functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL,
- C, C++, JAVA, python etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

6.2 Implementation Code

1) User Registration:

```
def register(request):
    if request.method=="POST":
        username = request.POST['username']
        email = request.POST['email']
        first_name=request.POST['first_name']
        last_name=request.POST['last_name']
        password = request.POST['password']
        confirm_password = request.POST['confirm_password']
        branch=request.POST['branch']
        Value = {
               'username':username,
                'firstname':first_name,
                'lastname':last_name,
                'email':email,
                'branch':branch
            }
        regex = re.compile('[@_!#$%^&*()<>?/\|}{~:]')
        atSymbole = email.index('@')
        dot =email.rfind(".")
        error_mess=""
        if User.objects.filter(username=username).exists():
            error_mess ="This User name have been taken Please try other"
        if User.objects.filter(email=email).exists():
            error_mess ="This email id is already exist"
        if len(username) > 15:
              error_mess = "username should be under 15 charecter"
        if(regex.search(username) != None):
                 error_mess = "username must be in alphabets"

        if(regex.search(first_name) != None):
            error_mess = "Name must be in alphabets"
        if password != confirm_password:
              error_mess = "Passwords do not match."
        if  dot != -1:
            if  (dot <= atSymbole + 3):
                error_mess = " Please check Your typed email, 3 or 4
                             character  required before dot"
            if  (dot != len(email)-4 and dot != len(email)-3):
                error_mess = " Please check Your typed email, After dot, 3 or
                               2 character required"
        else:
```

```
            error_mess = "Please check Your typed email, dot is required
                          after  @"

        if not error_mess:
         user = User.objects.create_user(username, email, password)
         Register.objects.create(user=user,branch=branch)
         user.first_name = first_name

         user.last_name = last_name
         user.save()
         return render(request, 'user_login.html')

        else:
            context={
                'error':error_mess,
                'values':Value
              }
            return render(request,"users/register.html",context)
    return render(request,"users/register.html")
```

2) User Login:

```
   def user_login(request):
       if request.method=="POST":
           username = request.POST['username']
           password = request.POST['password']
           user = authenticate(username=username, password=password)
           if user is not None:
               login(request, user)
               messages.success(request, "Successfully Logged In")
               return redirect("/user_home")
           else:
               messages.error(request, "Invalid Credentials")

       return render(request,"user_login.html")
```

3) Upload Notes:

```
@login_required
def upload_notes(request):
    user = User.objects.get(id=request.user.id)
    data = Register.objects.get(user = user)
    if request.method=='POST':
        branch = request.POST['branch']
        subject = request.POST['subject']
```

```
        notes = request.FILES['notesfile']
        filetype = request.POST['filetype']
        description = request.POST['description']

        user = User.objects.filter(username=request.user.username).first()
        Upload_Notes.objects.create(user=user,uploadingdate=date.today(),
        branch=branch,subject=subject,notesfile=notes,
        filetype=filetype,description=description,status='pending')
        messages.success(request,f"Notes uploaded from {request.user.username}
        successfully!")
        return redirect('/view_mynotes')
    return render(request,'user/upload_notes.html',{'branch':data.branch})
```

4) Post Question:

```
def forum(request):
    user = request.user
    profile = Profile.objects.all()
    user = User.objects.get(id=request.user.id)
    data = Register.objects.get(user = user)
    user_posts = Post.objects.filter(branch=data.branch).count()
    if request.method=="POST":
          user = request.user
          image = request.user.profile.image
          content = request.POST.get('content','')
          branch = request.POST.get('branch','')
          post = Post(user1=user, post_content=content,
          image=image,branch=branch)
          post.save()
          messages.success(request, f'Your Question has been posted
          successfully!!')
          return redirect('/forum')
    posts = Post.objects.filter(branch=data.branch,).order_by('-timestamp')
    print(data.branch)
    context={
        'posts':posts,
        'branch':data.branch,
        'user_posts':user_posts
    }
    return render(request, "forum.html",context)
```

5) Post Reply:

```
@login_required
def discussion(request, myid):
    post = Post.objects.filter(id=myid).first()
    replies = Replie.objects.filter(post=post)
    if request.method=="POST":
        form=ReplyContent(request.POST)
        if form.is_valid():
          user = request.user
          image = request.user.profile.image
          desc = request.POST.get('reply_content','')
          post_id =request.POST.get('post_id','')
          reply = Replie(user = user, reply_content = desc, post=post,
          image=image)
          reply.save()
          messages.success(request, f'Your Reply has been posted
           successfully!!')
          return redirect(f'/discussion/{post_id}')
        else:
            form=ReplyContent()
    form= ReplyContent()

    return render(request, "discussion.html", {'post':post,
         'replies':replies,'form':form})
```

6) Create Notes:

```
@login_required
def notes(request):
    if request.method=="POST":
         form=NotesForm(request.POST)
         if form.is_valid():
             notes=Notes(user=request.user,title=request.POST['title'],descrip
tion=request.POST['description'])
             notes.save()
             messages.success(request,f"Notes added successfully!")
         else:
            form=NotesForm()
    form=NotesForm()
    notes = Notes.objects.filter(user = request.user)
    context={'notes':notes,'form':form}
    return render(request,'user/notes.html',context)
```

7) Language Translator:

```
@login_required
def texttotxt(request):
     music = ''
     result=''

     if request.method == "POST":
        try:
          lang = request.POST.get("lang", None)
          text = request.POST.get("text", None)
          value={
              'text':text,
              'lang':lang
          }
          print(value)
          translator = Translator()
          tr = translator.translate(text, dest=lang)
          result=tr.text
          object = gTTS(text=result, lang=lang,slow=False)
          object.save("static/speech.mp3")
          music = "ok"
          context={
            'music':music,
            'result':result,
            'value':value
         }
          return render(request,'user/texttotxt.html',context)
        except Exception as e:
            messages.success(request, 'Something Went Wrong Try Again')
     else:
        pass
     context={
        'music':music,
        'result':result,

    }
     return render(request,"user/texttotxt.html",context)
```

Chapter 7. Testing

7.1 Psychology of testing:

Psychology of testing is a type of testing which fully depends on the mindset of developers and tester. When we are building the software, we working positively towards the software never think about negative things. The mindset should be different while testing and reviewing developing software. With the correct mindset, the programmer can test their own code. At a certain point independence tester often makes the tester more effective to finding defects.

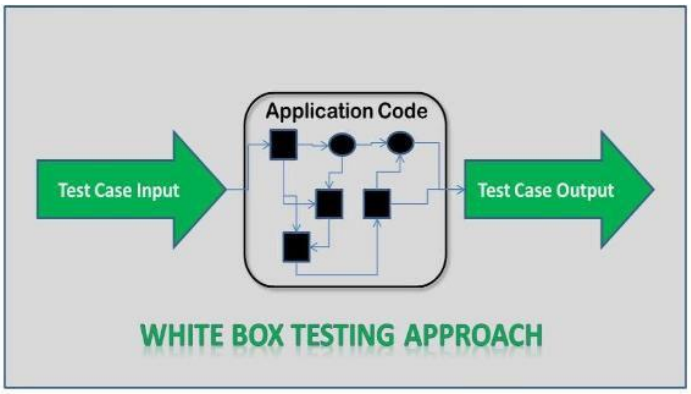
Levels of Independence Tester in Psychology of testing

- Tests designed by the person who wrote the software under test
- Tests designed by another person
- Tests designed by a person from a different organizational group or test specialists
- Tests designed by a person from a different organization or company

White box Testing sometimes called "Glass box testing" is a test case design uses the control structure of the procedural design to drive test case. Using white box testing methods, the following tests were made on the system

a) All independent paths within a module have been exercised once. In our system, ensuring that case was selected and executed checked all case structures. The bugs that were prevailing in some part of the code where fixed

b) All logical decisions were checked for the truth and falsity of the values.



Black box testing focuses on the functional requirements of the software. This is black box
testing enables the software engineering to derive a set of input conditions that will fully exercise
all functional requirements for a program. Black box testing is not an alternative to white box
testing rather it is complementary approach that is likely to uncover a different class of errors
that white box methods like.
1) Interface errors
2) Performance in data structure
3) Performance errors
4) Initializing and termination errors

7.2 Testing Objectives:

A TEST PLAN is a detailed document that describes the test strategy, objectives, schedule, estimation and deliverables and resources required for testing. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process which is minutely monitored and controlled by the test manager.

7.3 Levels of Testing:

7.3.1 Unit testing

Unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use.

A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure.

Ideally, each test case is independent from the others: substitutes like method stubs, objects, fakes and test harnesses can be used to assist testing a module in isolation.

7.3.2 Integration Testing:

This testing is sometimes called Integration and Testing. Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing.

Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates and delivers as its output the integrated system ready for system testing.

7.3.2 System Testing:

System Testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system.

Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

7.3.3 Validation Testing:

Validation Testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can reasonably expected by a customer.

After validation test has been conducted, one of the following two possible conditions exists. The functions or performance characteristics confirm to specification and are accepted.

• In the administrator modules, username must be unique.

• In the user registration, username and email must be unique.

7.3.4 User Acceptance Testing:

User acceptance of a system is a key factor of any system. The system under consideration is tested for the acceptance by constantly keeping in touch with the prospective system users at the same time of developing and marketing changes whenever required. This is done in regard to the following points:

• Input Screen Design

• Output Screen Design

7.4 test cases:

NO	INPUT GIVEN	EXPECTED OUTPUT	ACTUAL OUTPUT OCCURED	TEST PASS	ACTION TAKEN
1	Enter basic credentials, hit Register button	Success message, redirect login page	Error message	No	Email and username should be unique

2	username, pass	User Home page	User Home page	Yes	-
3	email	Send Reset Password link to user email	Email not received	No	Need to enter registered email
4	Username, pass	User Home page	Invalid Credentials	No	Need to check username and password.
5	Input Text, select language	Show translated text and audio.	Error message	No	Need to check device internet connection.
6	Enter book search	Show a list of search related book	Books does not show	No	Need to enter correct book name

Chapter 8: Conclusion

Finally, it can be said that Study Portal is really helpful in the study as It provides managed environment containing amazing features which allow the user to access study notes uploaded by others and user also can share their own notes and discuss queries related to their stream and the user will be able to use all other features like create notes, to-do, language translator, book search functionality, and pdf reader according to their need. It is not intended for a particular organization this project going to develop generic software, which can be used by any Study organization.

8.1 The Obstacles:

• Working with Django framework & Mysql Server is new experience for me.

•I adopted these things by video tutorials, text tutorials, internet and learning materials given by the tools themselves. It's a matter of time, patience and hard work.

• It is very sensible work and it demands much time.

8.2 The Achievements:

It has been a great pleasure for me to work on this exciting and challenging project.

This project proved good for me as it provided practical knowledge of not only programming in Python Django based application and MySQL Server, but also about all handling procedure related with “Study Portal”. It also provides knowledge about the latest technology used in developing such applications which will be great a demand in near future. This will provide better opportunities and guidance in future in developing projects independently.

8.3 Future Plans:

This Study Portal involves many more features related to study with their best performance.

The future implementation will be adding some extra features and branches in the portal and adding some extra features in the forum section

Chapter 9: Bibliography

References:

1. https://stackoverflow.com/
2. http://www.youtube.com/
3. https://docs.djangoproject.com/en/4.0/
4. https://www.w3schools.com/
5. https://www.google.com