Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech. CE Semester – IV

Subject: Software Engineering

Project title: Knowledge Market

By:

1) Smit Bhoraniya, Roll no: CE012 , Id: 19CEUOS057

2) Charvit Ardeshna, Roll no: CE005 , Id: 19CEUOS130

Guided by: Prof. Jigar M. Pandya

Prof. Pinkal C. Chauhan

Prof. Brijesh S. Bhatt

Contents

Abstract

It is for students/learners, with the ultimate intention of collectively increasing good knowledge in the world. No matter what problem you have face No matter what subject you can get knowledge by any mean. "Every Problem has solution“

‘Lets take step ahead to making the internet better place’

Technologies/tools used

Technologies:

Django , Python , dbSqlite, Bootstrap , HTML5 , CSS3, JS

Tools:

Git , Visual Studio Code, Sublime text3, Pycharm

Platform:

Local development server(Soon Heroku)

Software Requirements Specifications

1. Admin module:

1. Verify admin:

Description:- Admin can needs to enter their details and verify themselves.

Input:-Information of admin.

2. User module

1. Registration:

Description:-Take details of any user such as name, mobile number, email-id, Profile picture etc. That data stored in database and Account for user is created.

Input:-Information of users.

Output:-Create Account

1.1 Update profile:-

Description:-A user can change his details and update it like Profile picture, mobile no., email-id, name etc.
Input:-Persons details
Output:- Account updated

1.2 Display current details:-

Input:-User's id
Output:-Users details

1.3 Delete Account:-

Description:-A user have some issue and he wants to create new profile and delete this old one than he can delete profile .
Input:-User's id
Output:-Id is deleted

2. login:-

Description:-A user can input main details and can enter in website and can use it.
Input:-Username, Password
Output:-Homepage of website

3. Questions Module:-

1. Post Question:-

Description:-A user have doubt on any special topic he can ask his question and post it that all other user can see that. In this user have to fill the form in which he have to put title of Question and its description.

Input:-Question Details
Output:- All user can see(Post on the page of website)

2. display Questions:-

Description:-A user can see all the questions which are posted by other users.
Output:-Page of questions.

3. delete Questions:-

Description:-A user want to delete his question on some reason than he can delete it and upload a new question.

4. Searching Module:-

1. search account:-

Description:-A user can search any account for get some info about user.
Input:-user id.
Output:-User Profile

2. search question:-

Description:-A user can search specific question which he wants tosolve in specific tag.
Input:-Title of question
Output:-Questions

5. Answers Module:-

1. Reply to the post:-

Description:-A user can see the questions and if he knows the answer than he can reply that post.
Input:-Answer
Output:-All user can see(Post on the page of website)

1.1 Display Answers:-

Description:-A user can see all the answers of thequestion which are posted by other users.
Output:-Page of questions

1.2 delete Answers:-

Description:-A user want to delete his answer on somereason like he know that his answer was slightly wrong than he can delete it and upload a new answer.

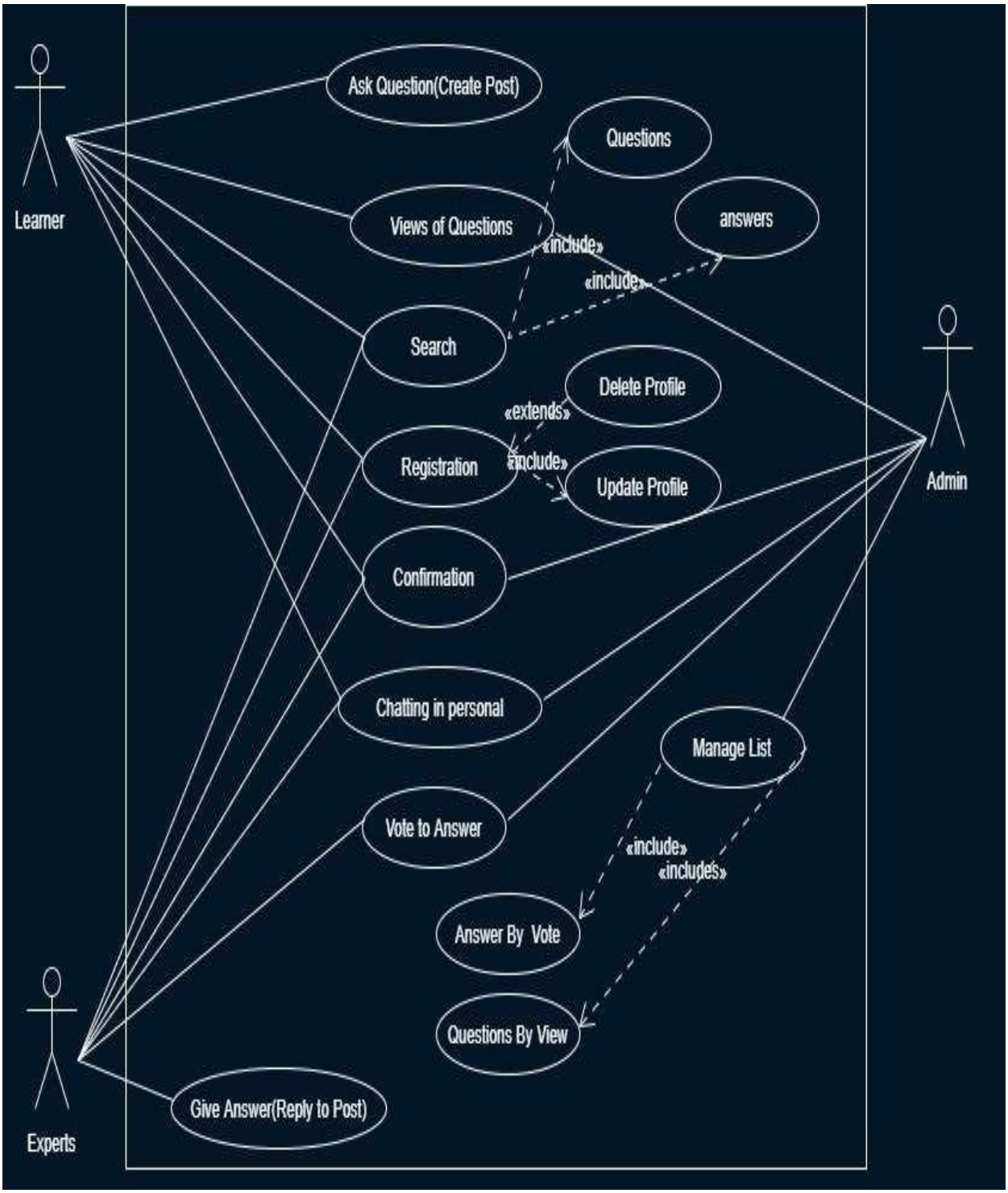
6. Like Module:

1.Like Answer and Question:-

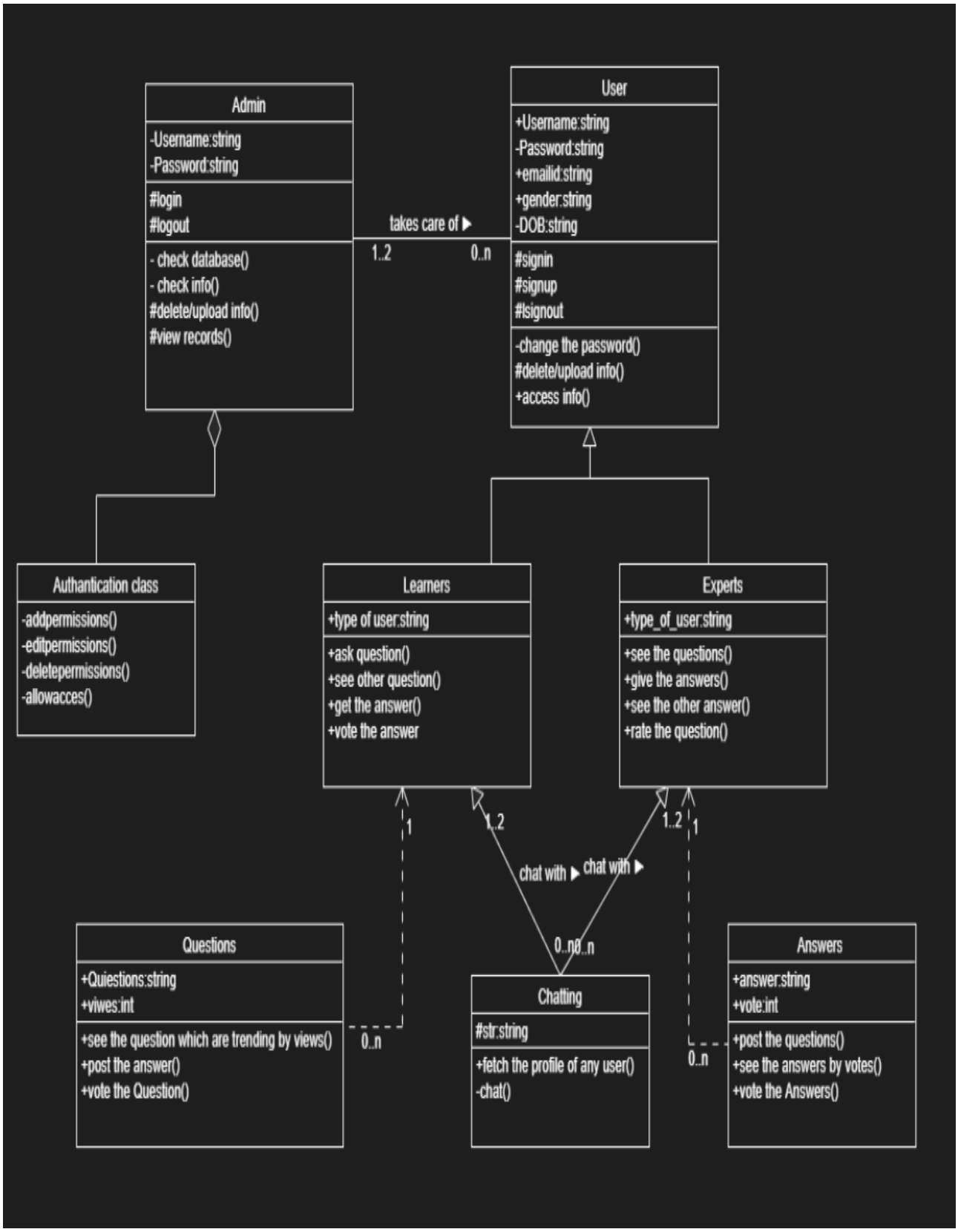
Description:-user can like answers and also questions
Input:-like
Output:-Increase the like of question or answer also get notification

Design

1. Use case diagram:

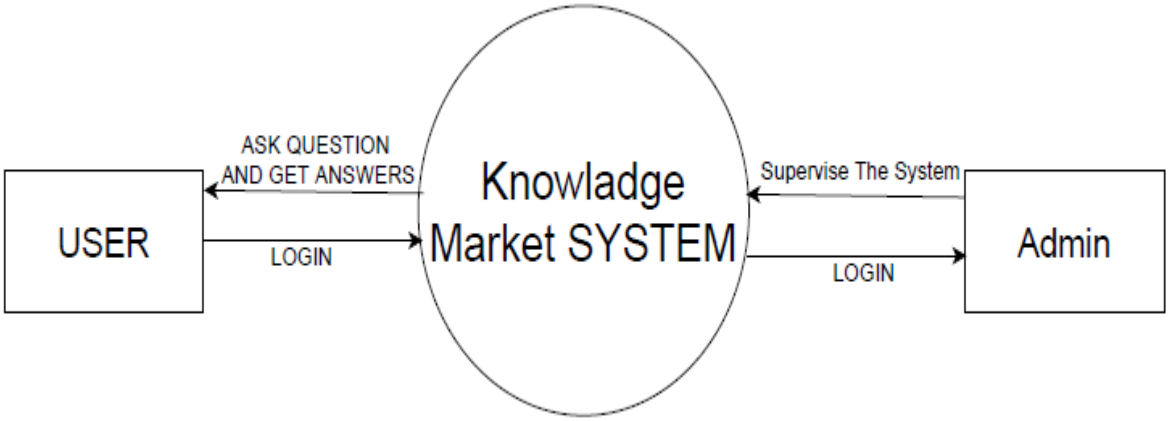


2. Class Diagram:

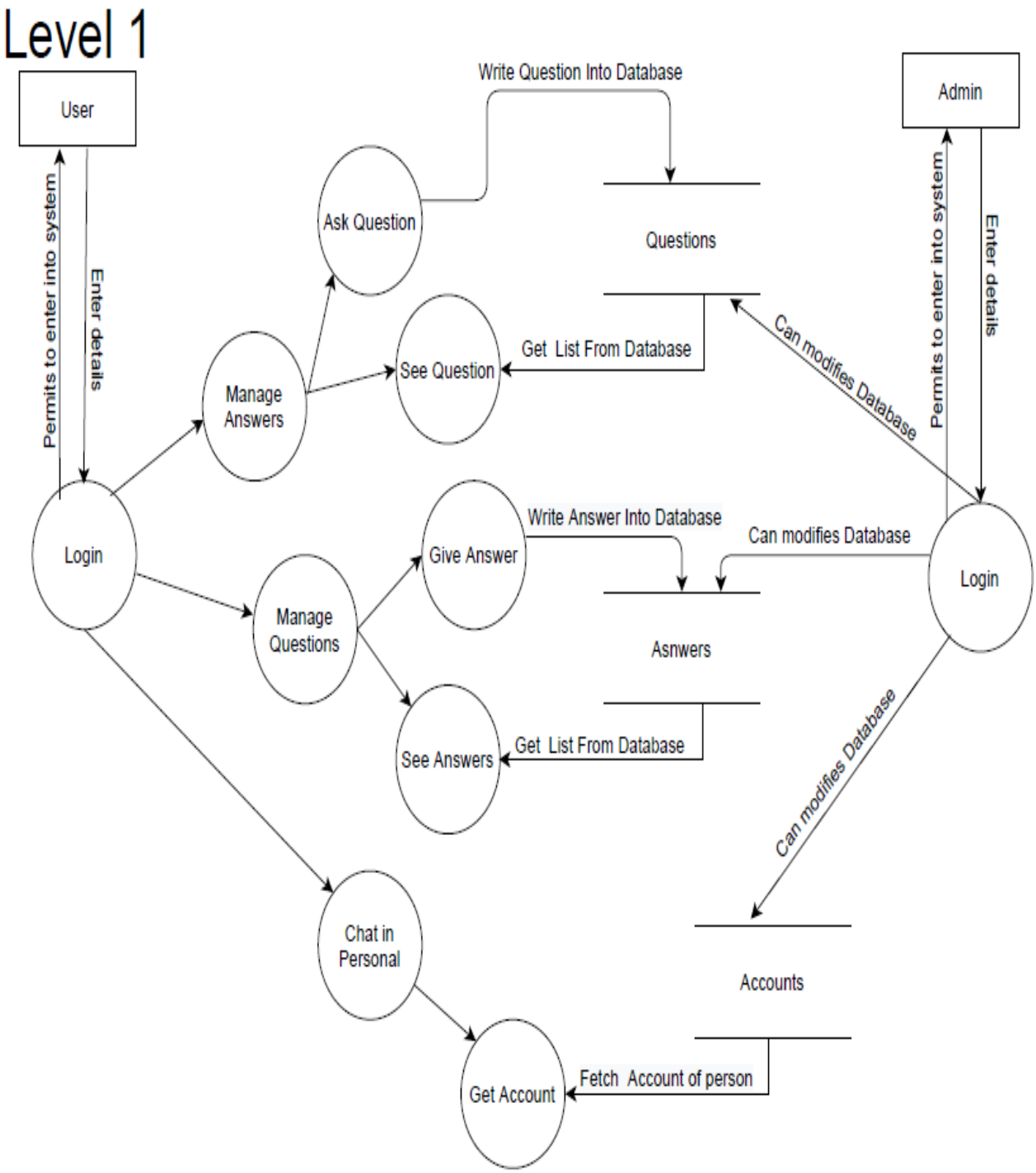


3. DFD Model

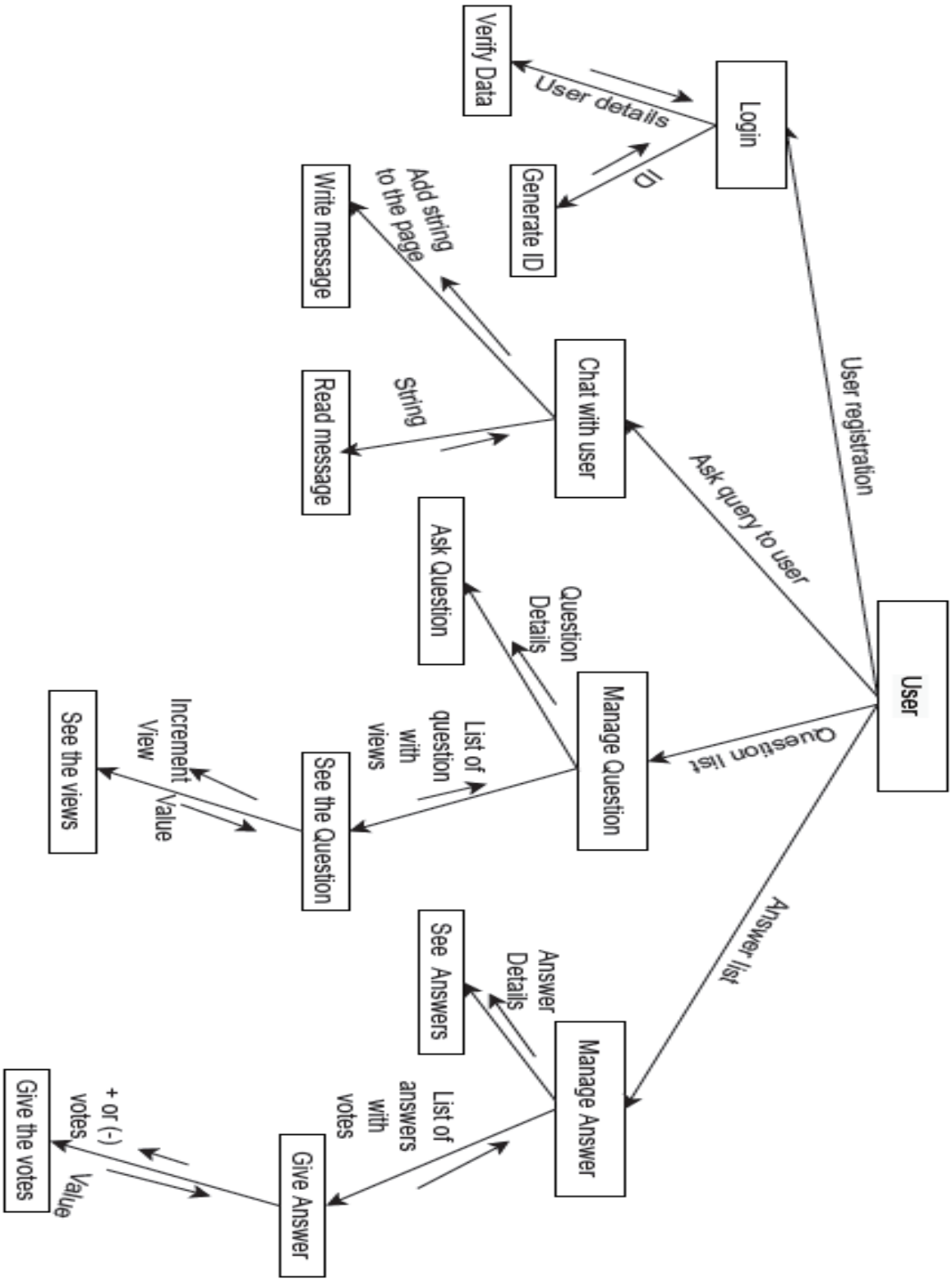
I. Level 0 : Context Diagram



II. Level 1 : Context Diagram

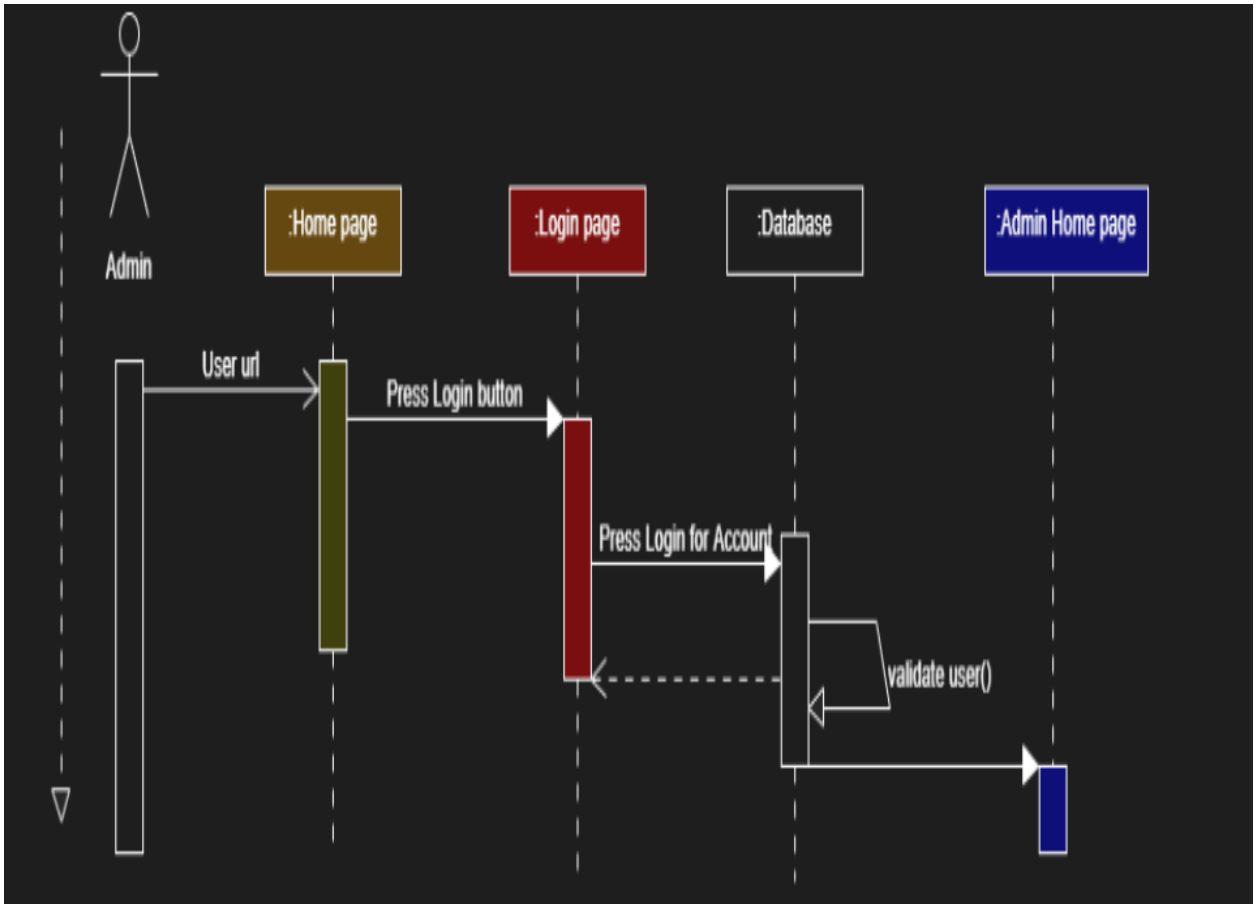


4. Structure chart

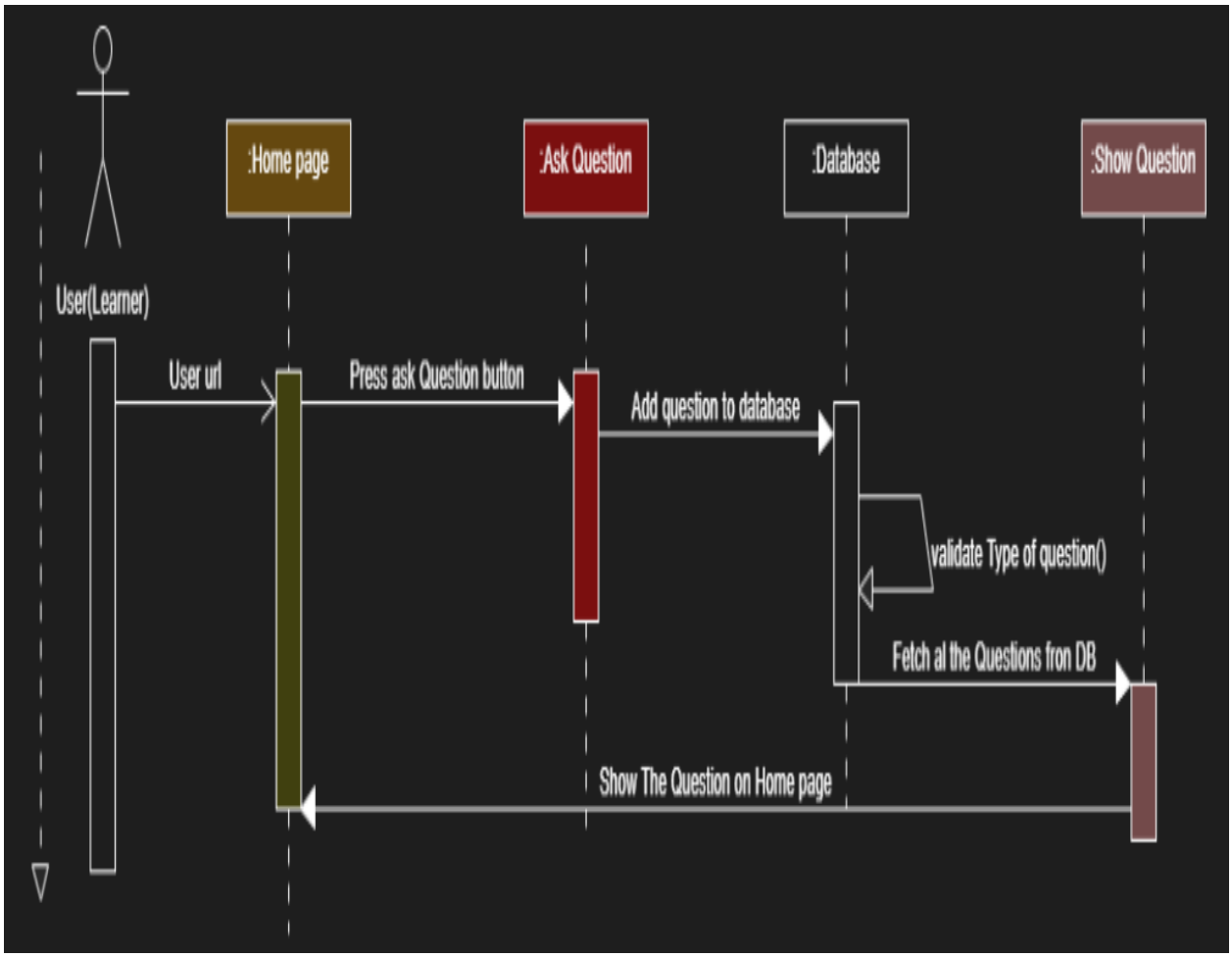


5. Sequence Diagram:

I. Login System

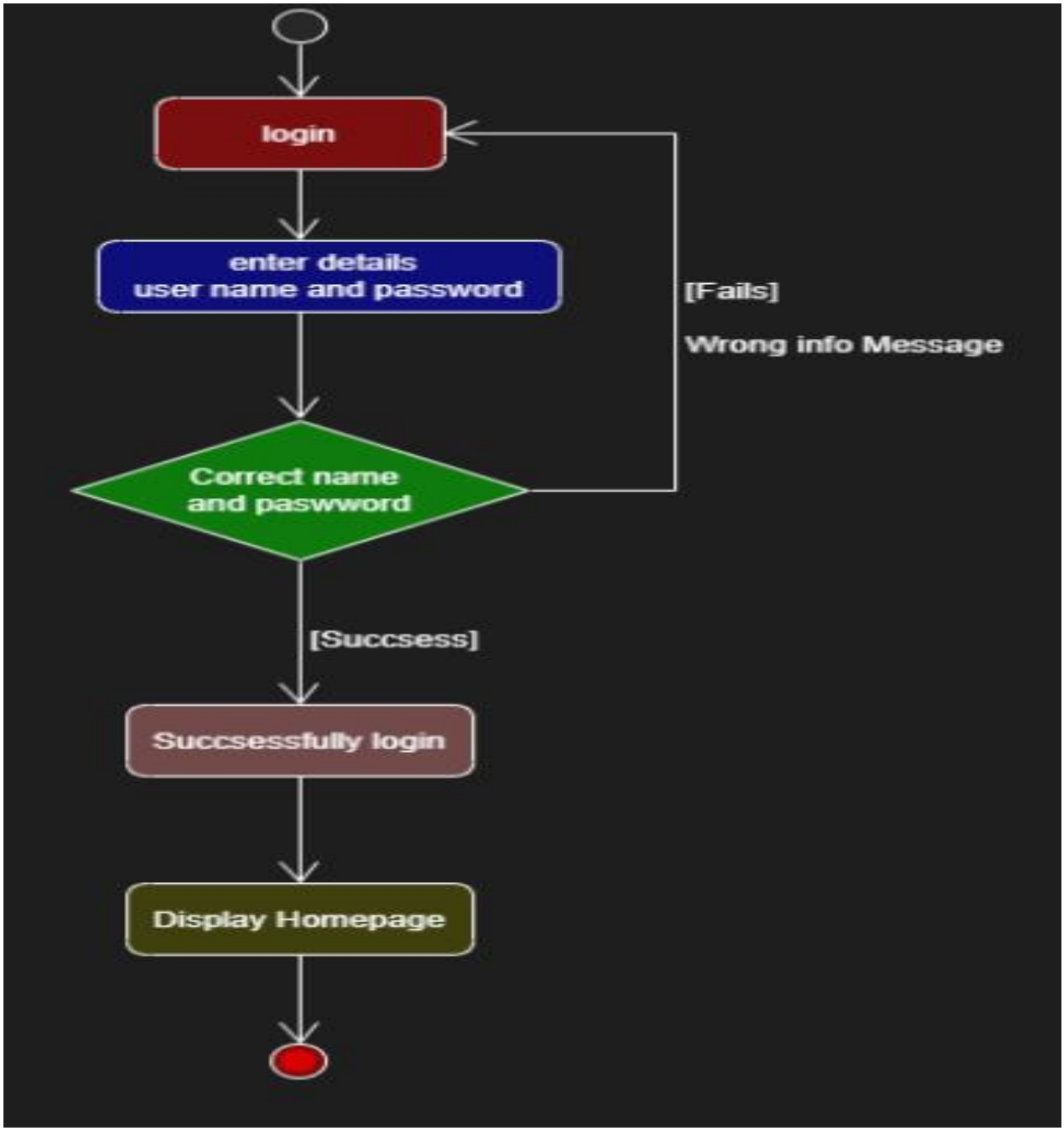


II. Ask Question

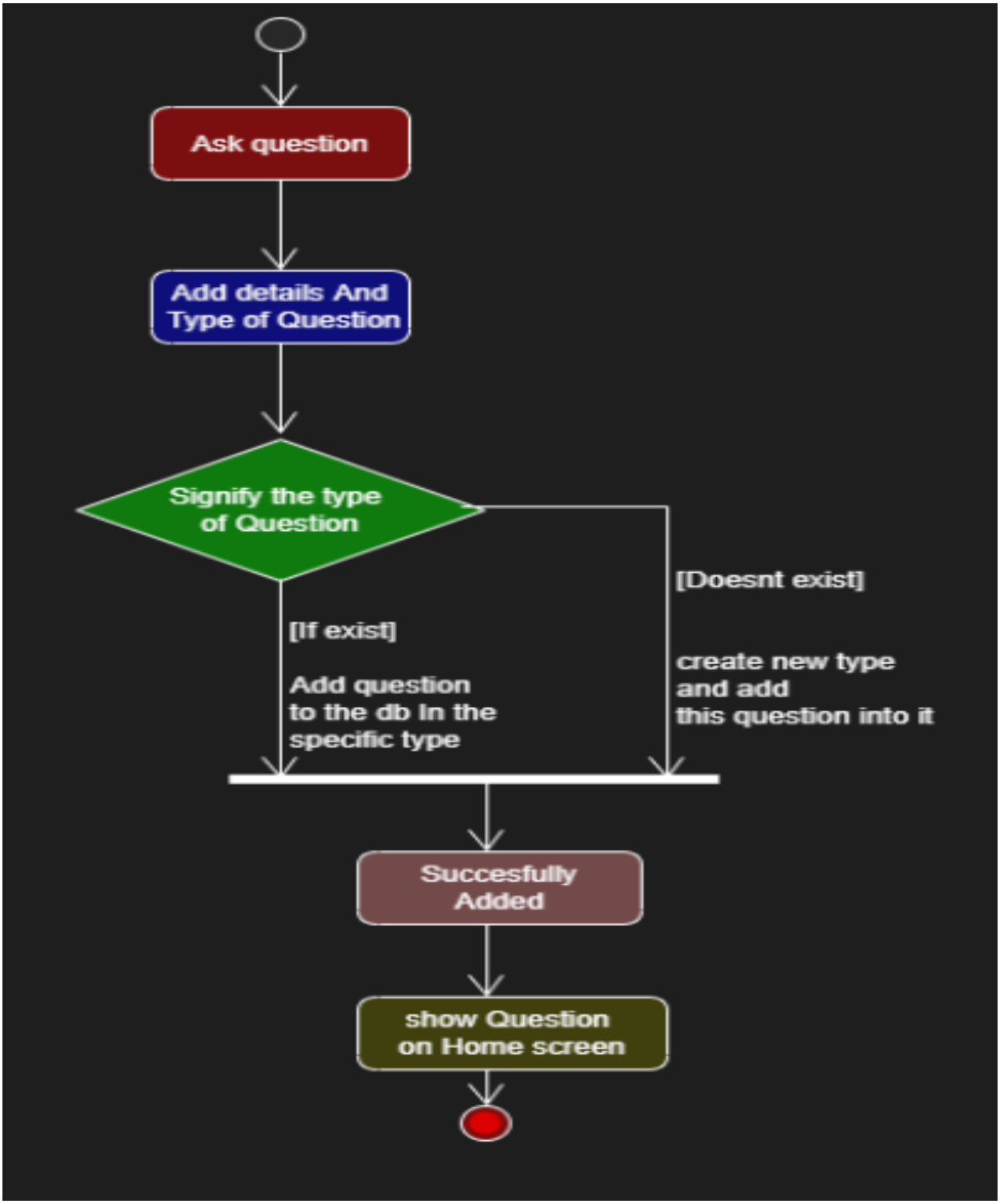


6. Activity Diagram:

I. Login:



II. Ask Question



Implementation Detail

1. Modules:

In the following section a brief description of each module is given. Related screenshots are attached in separate sections.

The system consists of 5 basic modules namely

I. Admin Module

II. User Module

III. Question Module

IV. Answer Module

V. Notification Module

Each module consists of several methods to implement the required functionality. Implementation is done using Django. Database used in these modules is DBSqlite.

I. Admin Module

Admin module can modify user requirment. Admin can modify any question or answer which is posted by users. Admin can also remove users.

II. User Module

User can sign up to the system and then login. User can ask question, give answer of other user's question , if user forgot password then he can change password via Gmail, if he want then he can modify profile

III. Question Module

A user have doubt on any specific topic he can ask his question and post it that all other user can see that. In this user have to fill the form in which he have to put title of Question and its description. If User find helpful question then he can like that question, if user want to update or delete his question then he can

IV. Answer Module

A user can see the questions and if he knows the answer than he can reply that post. If User find helpful Answer then he can like that Answer, if user want to update or delete his Answer then he can.

V. Notification Module

If any user like question or answer which is yours then it will notify you, and if any user commented on your question then it will notify you

2. Major Functions prototypes

Login:

User needs to provide their username and password .If it matches to database then Users can login to the system.

```
def login_view(request, *args, **keargs):
    context = {}
    user = request.user
    if user.is_authenticated:
        return redirect("account:home")

    if request.POST:
        form = AccountAuthenticationForm(request.POST)
        if form.is_valid():
            email = request.POST['email']
            password = request.POST['password']
            user = authenticate(email=email, password=password)
            if user:
                login(request, user)
                destination = get_redirect_if_exist(request)
                if destination:
                    return redirect(destination)
                return redirect("account:home")
        else:
            context['login_form'] = form
    return render(request, "account/login.html", context)
```

Registration:

If user is new to the system he needs to fill registration form And details of form will be stored to database.

```
def register_view(request, *args, **kwargs):
    user = request.user
    if user.is_authenticated:
        return HttpResponse(f"You are already authenticated as {user.email}.")

    context = {}
    if request.POST:
        form = RegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            email = form.cleaned_data.get('email').lower()
            raw_password = form.cleaned_data.get('password1')
            account = authenticate(email=email, password=raw_password)
            login(request, account)
            destination = get_redirect_if_exist(request)
            if destination:
                return redirect(destination)
            return redirect('account:home')
        else:
            context['registration_form'] = form
    return render(request, 'account/register.html', context)
```

Contact Us – About Us:

Users can Contact to admin via this form

User can get info about site by about us page

```
def contect_view(request):
    form = ContectUsForm()
    context = {}
    if request.POST:
        form = ContectUsForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('account:home')
        else:
            context['form'] = form
    return render(request, 'account/contect_us.html')

def about_us_view(request):
    return render(request, 'account/aboutus.html')
```

Notification:

```
def ShowNOtification(request, *args, **kwargs):
    user = request.user
    context = {}
    if not user.is_authenticated:
        return redirect("account:login")
    notifications = Notification.objects.filter(
        receiver=user).order_by('-date')
    Notification.objects.filter(receiver=user, is_seen=False).update(is_seen=True)
    context = {'notifications': notifications}
    return render(request, 'notification/notify.html', context)

def DeleteNotification(request, *args, **kwargs):
    user = request.user
    noti_id = kwargs.get("noti_id")
    notification = Notification.objects.filter(id=noti_id)
    notification.delete()
    return redirect('notification:show_notification')


def CountNotification(request):
    count_notifications = 0;
    if request.user.is_authenticated:
        count_notifications = Notification.objects.filter(receiver=request.user, is_seen=False).count()

    return {'count_notifications': count_notifications}
```

Search:

User can search question or user profile.

```
def account_search_view(request, *args, **kwargs):
    context = {}
    account_nav = True
    context['account_search'] = account_nav
    if request.method == "GET":
        search_query = request.GET.get("q")
        if len(search_query) > 0:
            search_results = Account.objects.filter(email__icontains=search_query).filter(
                username__icontains=search_query).distinct()
            user = request.user
            accounts = []
            for account in search_results:
                accounts.append((account, False))
            context['accounts'] = accounts
            context['user'] = request.user
            context['search_query'] = search_query
    context['account_search'] = True
    context['search'] = True
    return render(request, "account/search_results.html", context)
```

```
def question_search_view(request, *args, **kwargs):
    context = {}
    if request.method == "GET":
        search_query = request.GET.get("q")
        if len(search_query) > 0:
            search_results = Question.objects.filter(
                title__icontains=search_query)
            user = request.user
            questions = []
            for question in search_results:
                questions.append(question)
            print(questions)
            context['questions'] = questions
    context['question_search'] = True
    context['search'] = True
    return render(request, "question/search_results.html", context)
```

Question:

in this function it fetches the data from database and show data to user and user can update or remove it.

```
def add_question(request, *args, **kwargs):
    context = {}
    user = request.user
    auth = Account.objects.get(id=user.id)
    if user.is_authenticated:
        form = AddQuestionForm()
        if request.POST:
            form = AddQuestionForm(request.POST)
            if form.is_valid():
                obj = form.save(commit=False)
                obj.auth = auth
                obj.save()
                return redirect('account:view', user_id=user.id)
            else:
                context['form'] = form
    context['CRUD'] = True
    return render(request, 'question/add_question.html', context)


def remove_question(request, *args, **kwargs):
    context = {}
    user = request.user
    question_id = kwargs.get("question_id")
    if user.is_authenticated:
        try:
            question = Question.objects.get(id=question_id)
        except Exception as e:
            return HttpResponse("Question not exists!!")

        context['question'] = question
        if request.POST:
            question.delete()
            return redirect('account:view', user_id=user.id)
    context['CRUD'] = True
    return render(request, 'question/remove_question.html', context)
```

```python
def edit_question_view(request, *args, **kwargs):
    user = request.user
    if not request.user.is_authenticated:
        return redirect("account:login")
    question_id = kwargs.get("question_id")
    question = Question.objects.get(pk=question_id)
    context = {}
    if request.POST:
        form = QuestionUpdateForm(
            request.POST, request.FILES, instance=question)
        if form.is_valid():
            obj = form.save(commit=False)
            obj.auth = user
            obj.save()
            return redirect("question:question", question_id=question.pk)
        else:
            form = QuestionUpdateForm(request.POST, instance=question,
                                      initial={
                                          "id": question.pk,
                                          "title": question.title,
                                          "question": question.question,
                                      }
                                      )
            context['form'] = form
    else:
        form = QuestionUpdateForm(request.POST, instance=question,
                                  initial={
                                      "id": question.pk,
                                      "title": question.title,
                                      "question": question.question,
                                  }
                                  )
        context['form'] = form
    context['CRUD'] = True
    return render(request, "question/edit_question.html", context)
```

Answer:

In this function it fetches the data from database and show data to user and user can update or remove it.

```
def add_answer(request, *args, **kwargs):
    context = {}
    user = request.user
    question_id = kwargs.get("question_id")
    auth = Account.objects.get(id=user.id)
    question = Question.objects.get(id=question_id)
    if user.is_authenticated:
        form = AddAnswerForm()
        if request.POST:
            form = AddAnswerForm(request.POST)
            if form.is_valid():
                obj = form.save(commit=False)
                obj.auth = auth
                obj.question = question
                obj.save()
                return redirect('question:question', question_id=question.id)
            else:
                context['form'] = form
    context['CRUD'] = True
    return render(request, 'question/add_answer.html', context)


def remove_answer(request, *args, **kwargs):
    context = {}
    user = request.user
    answer_id = kwargs.get("answer_id")
    question_id = kwargs.get("question_id")
    if user.is_authenticated:
        try:
            answer = Answer.objects.get(id=answer_id)
        except Exception as e:
            return HttpResponse("Answer not exists!!")

        context['answer'] = answer
        if request.POST:
            answer.delete()
            return redirect('question:question', question_id=question_id)
    context['question_id'] = question_id
    context['CRUD'] = True
    return render(request, 'question/remove_answer.html', context)
```

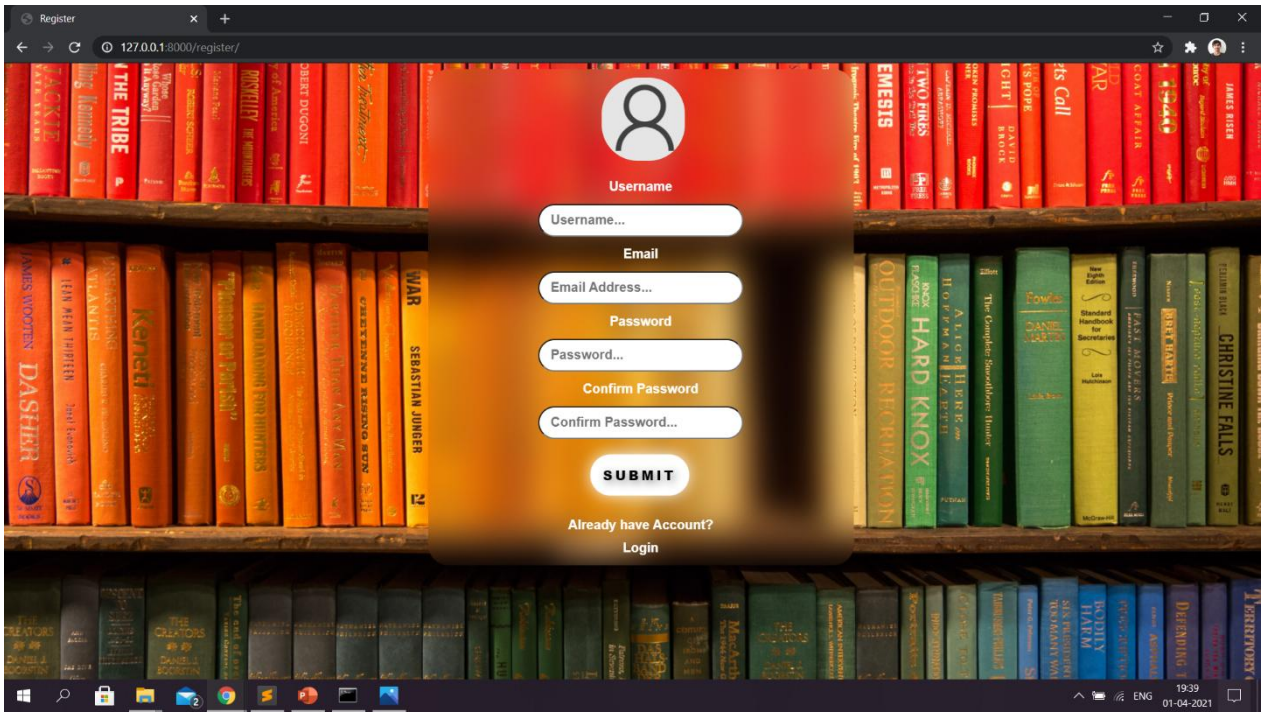
```python
def edit_answer_view(request, *args, **kwargs):
    user = request.user
    if not request.user.is_authenticated:
        return redirect("account:login")
    question_id = kwargs.get("question_id")
    answer_id = kwargs.get("answer_id")
    question = Question.objects.get(pk=question_id)
    answer = Answer.objects.get(pk=answer_id)
    context = {}
    if request.POST:
        form = AnswerUpdateForm(
            request.POST, request.FILES, instance=answer)
        if form.is_valid():
            obj = form.save(commit=False)
            obj.question = question
            obj.auth = user
            obj.save()
            return redirect("question:question", question_id=question.pk)
        else:
            form = AnswerUpdateForm(request.POST, instance=answer,
                                    initial={
                                        "id": answer.pk,
                                        "answer": answer.answer,
                                    }
                                    )
            context['form'] = form
    else:
        form = AnswerUpdateForm(request.POST, instance=answer,
                                initial={
                                    "id": answer.pk,
                                    "answer": answer.answer,
                                }
                                )
        context['form'] = form
    context['CRUD'] = True
    return render(request, "question/edit_answer.html", context)
```

Like:

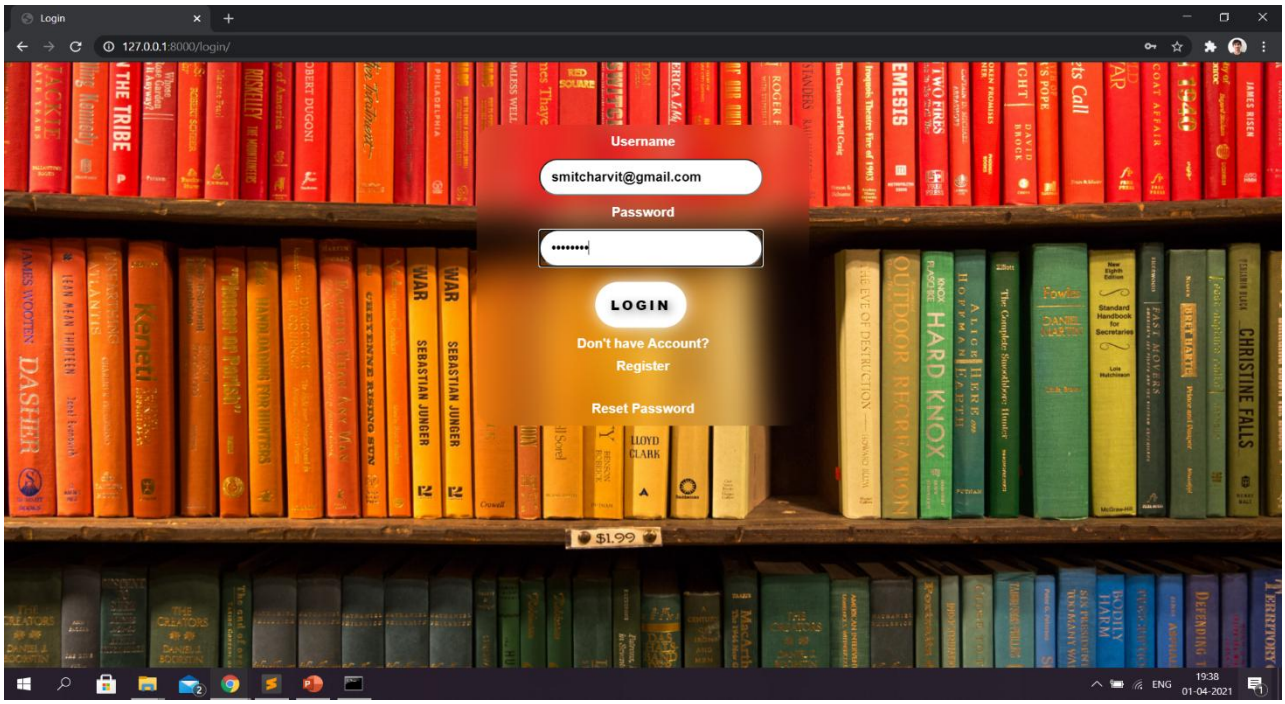
```
def like_question_view(request, *args, **kwargs):
    user = request.user
    question_id = kwargs.get("question_id")
    if user.is_authenticated:
        if request.POST:
            id = request.POST.get('question_like')
            obj = Question.objects.get(id=id)

            if user in obj.like.all():
                obj.like.remove(user)
            else:
                obj.like.add(user)

            liked, created = LikeQuestion.objects.get_or_create(
                auth=user, question_id=id)

            if not created:
                if liked.value == 'Like':
                    liked.value = 'unlike'
                else:
                    liked.value = 'like'
            liked.save()
    return redirect('question:question', question_id)


def like_answer_view(request, *args, **kwargs):
    user = request.user
    question_id = kwargs.get("question_id")
    if user.is_authenticated:
        if request.POST:
            id = request.POST.get('answer_like')
            obj = Answer.objects.get(id=id)

            if user in obj.like.all():
                obj.like.remove(user)
            else:
                obj.like.add(user)

            liked, created = Like.objects.get_or_create(auth=user, ans_id=id)

            if not created:
                if liked.value == 'Like':
                    liked.value = 'unlike'
                else:
                    liked.value = 'like'
            liked.save()
    return redirect('question:question', question_id)
```

Layout

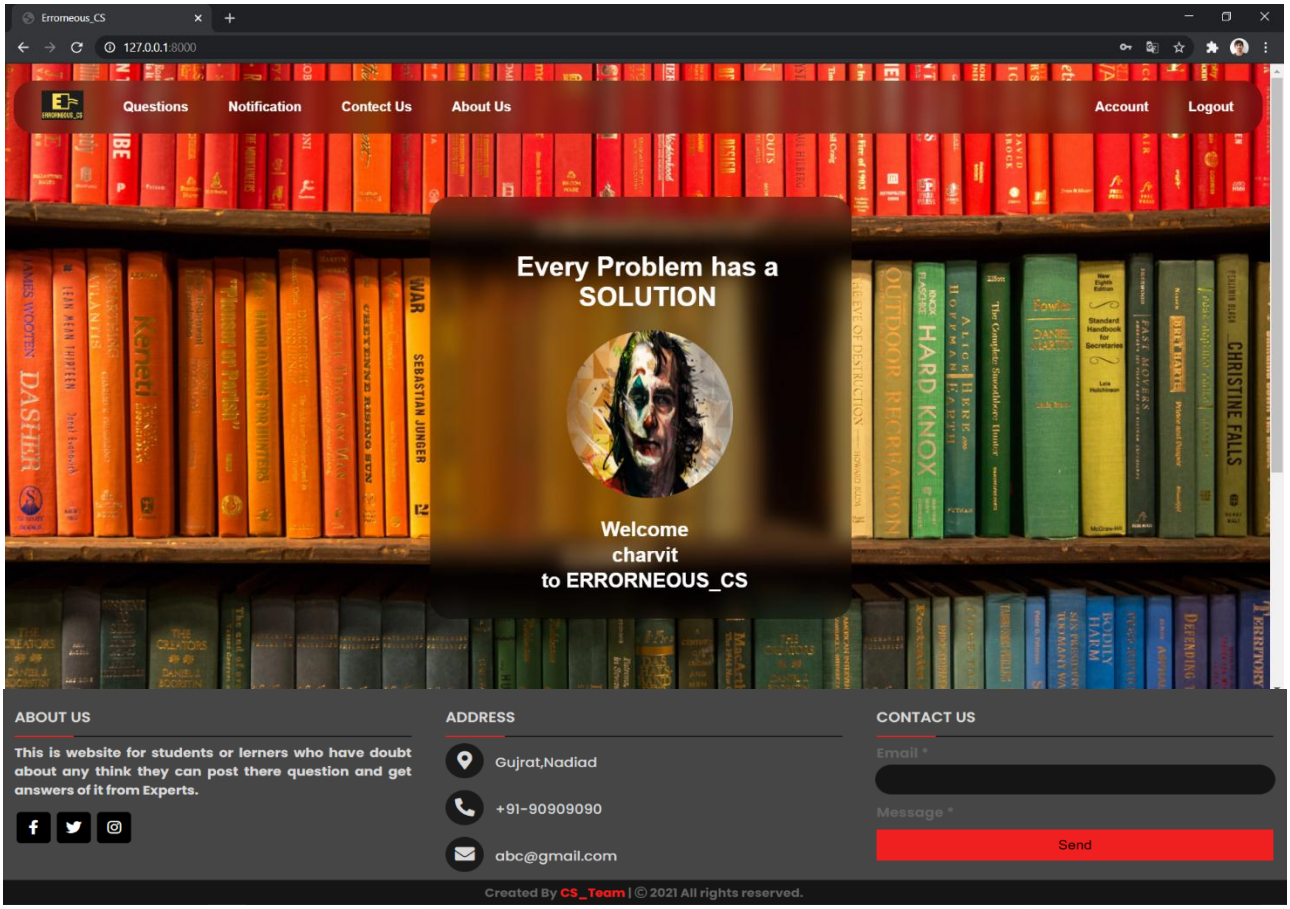
Signup Page



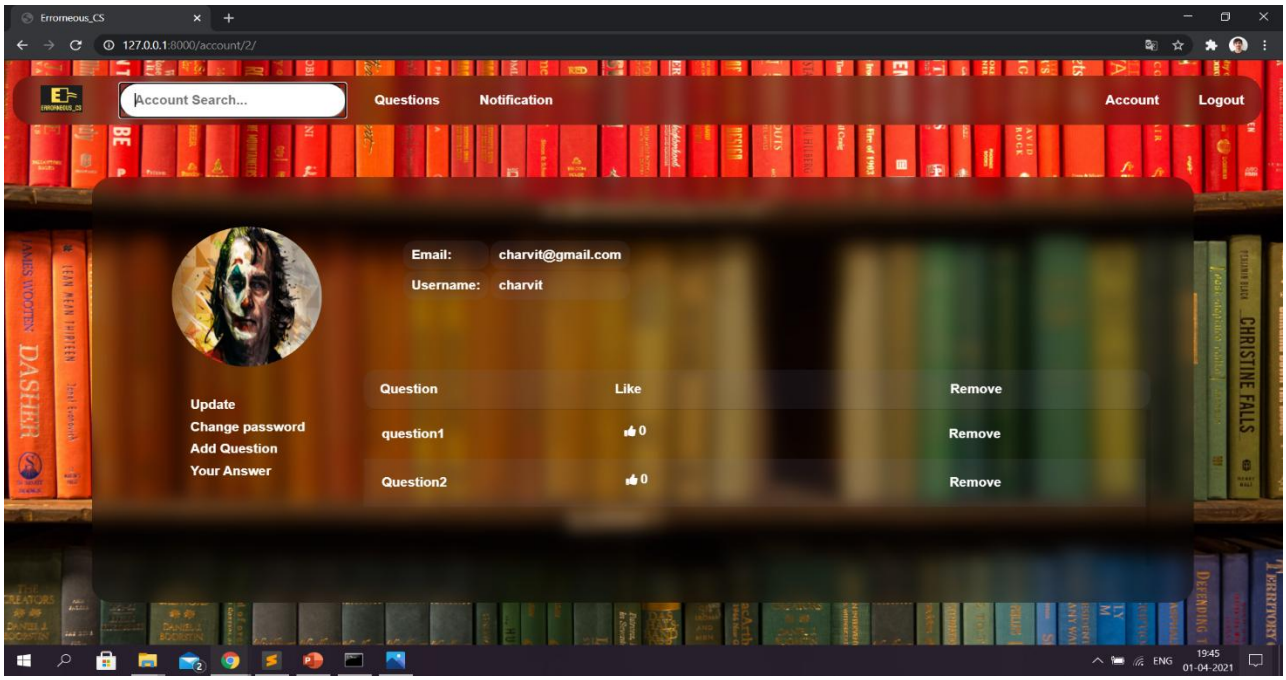
Login Page



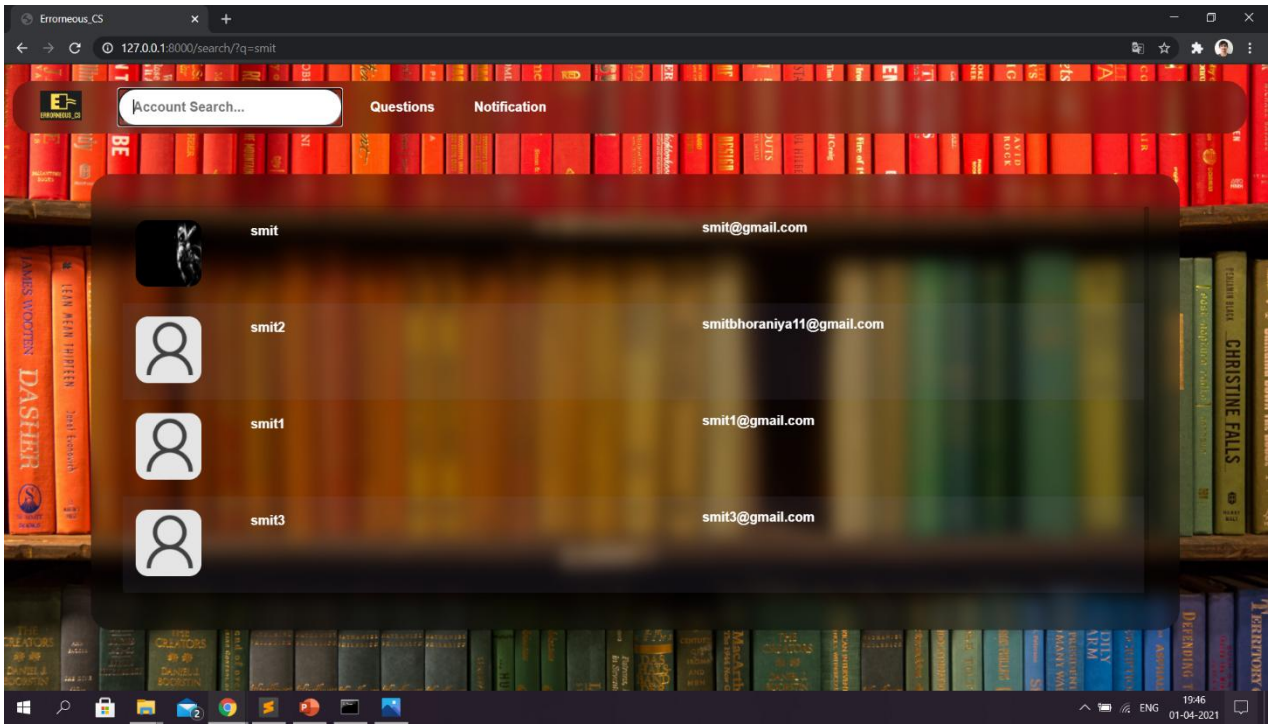
Home Page


Errorneous_CS
127.0.0.1:8000
Questions
Notification
Contect Us
About Us
Account
Logout
Every Problem has a SOLUTION
Welcome
charvit
to ERRORNEOUS_CS
ABOUT US
This is website for students or lerners who have doubt about any think they can post there question and get answers of it from Experts.
ADDRESS
Gujrat,Nadiad
+91-90909090
abc@gmail.com
CONTACT US
Email *
Message *
Send
Created By CS_Team | © 2021 All rights reserved.

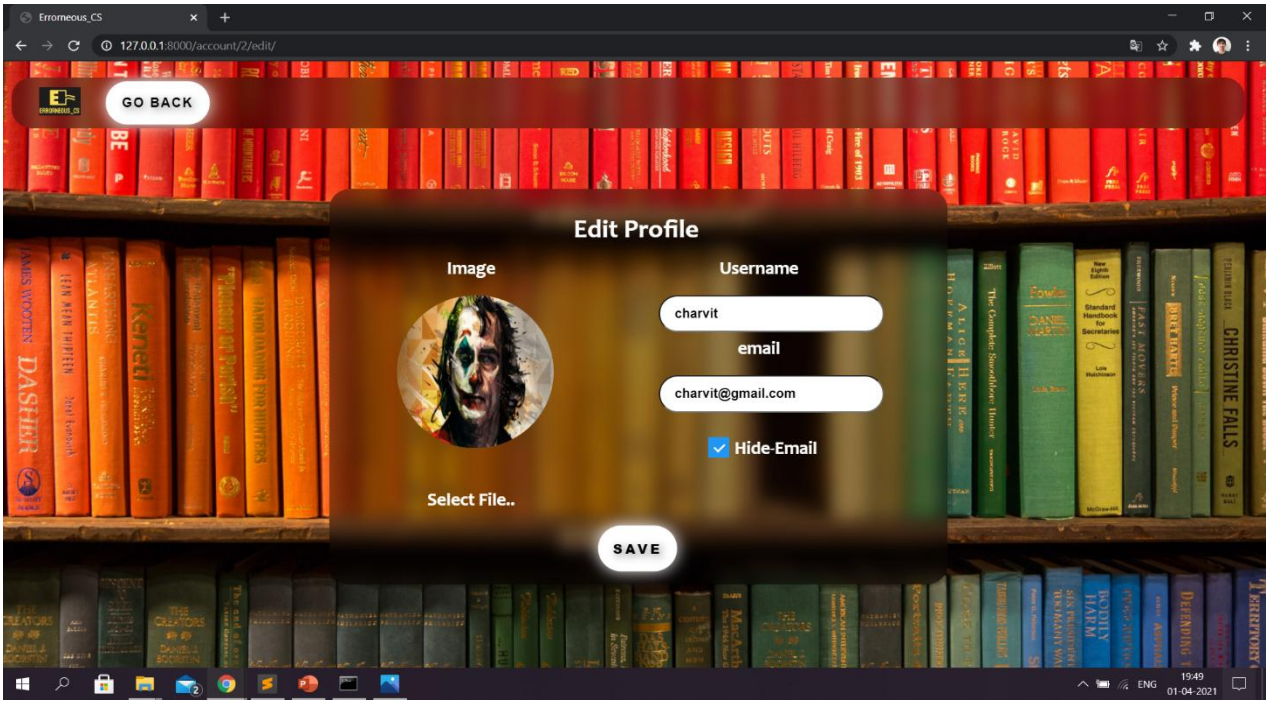
Account



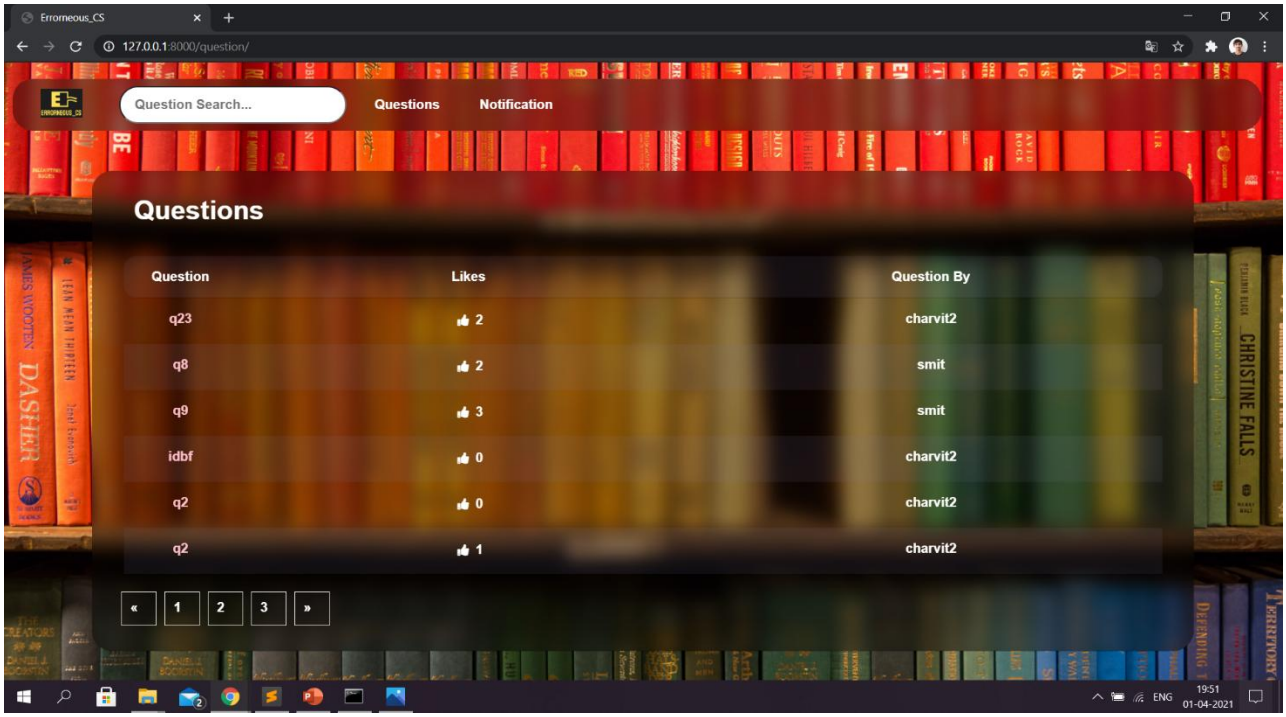
Search Account



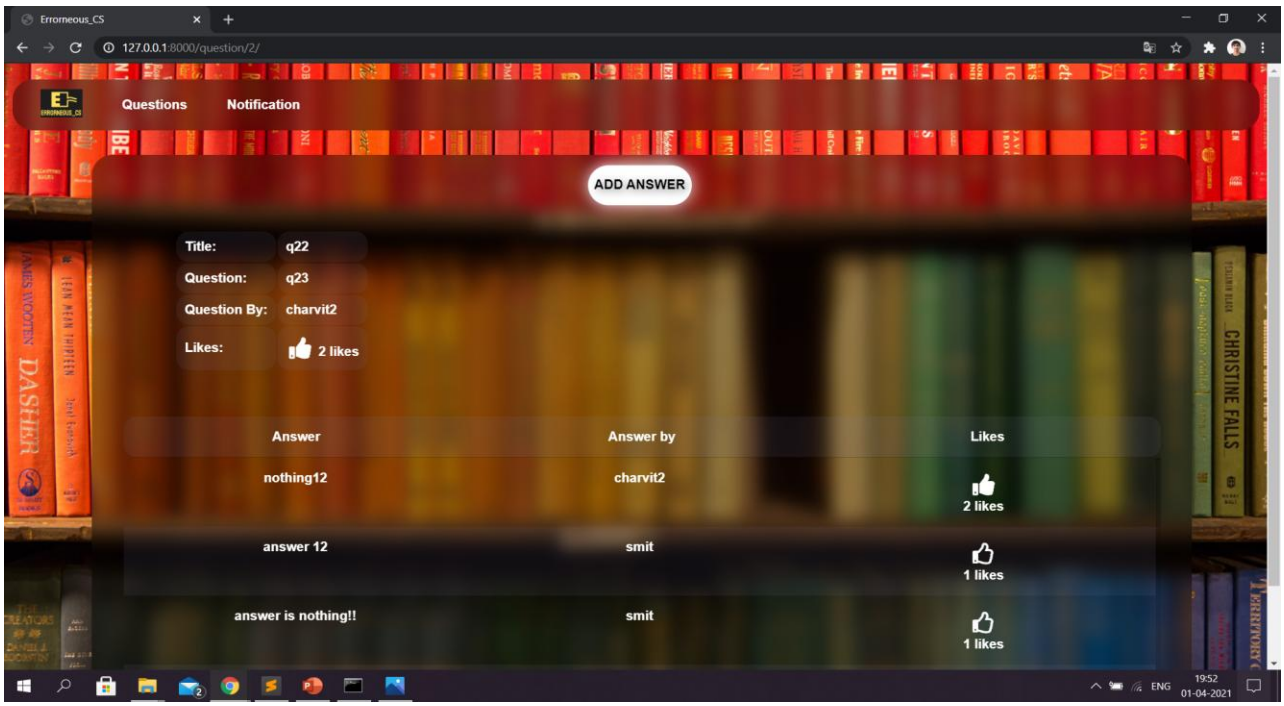
Update Account



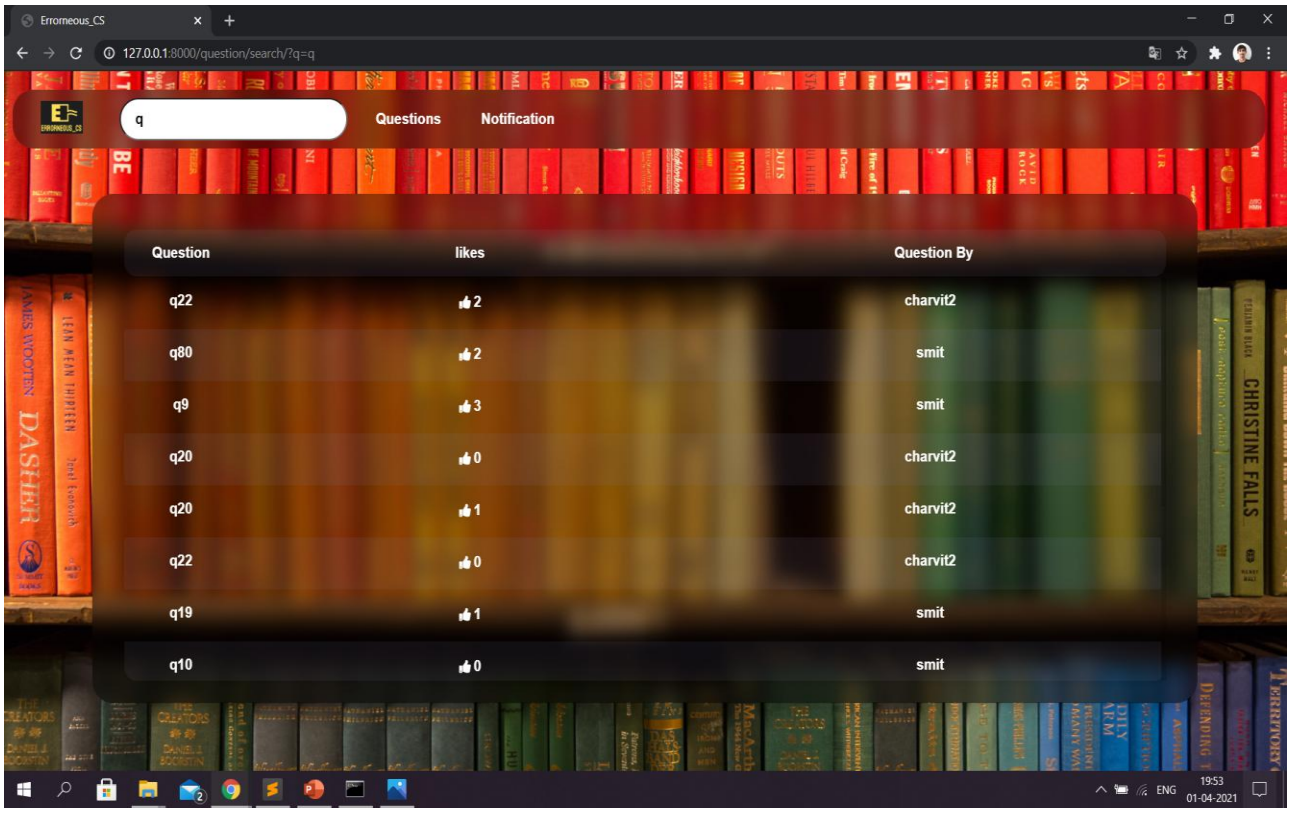
Question



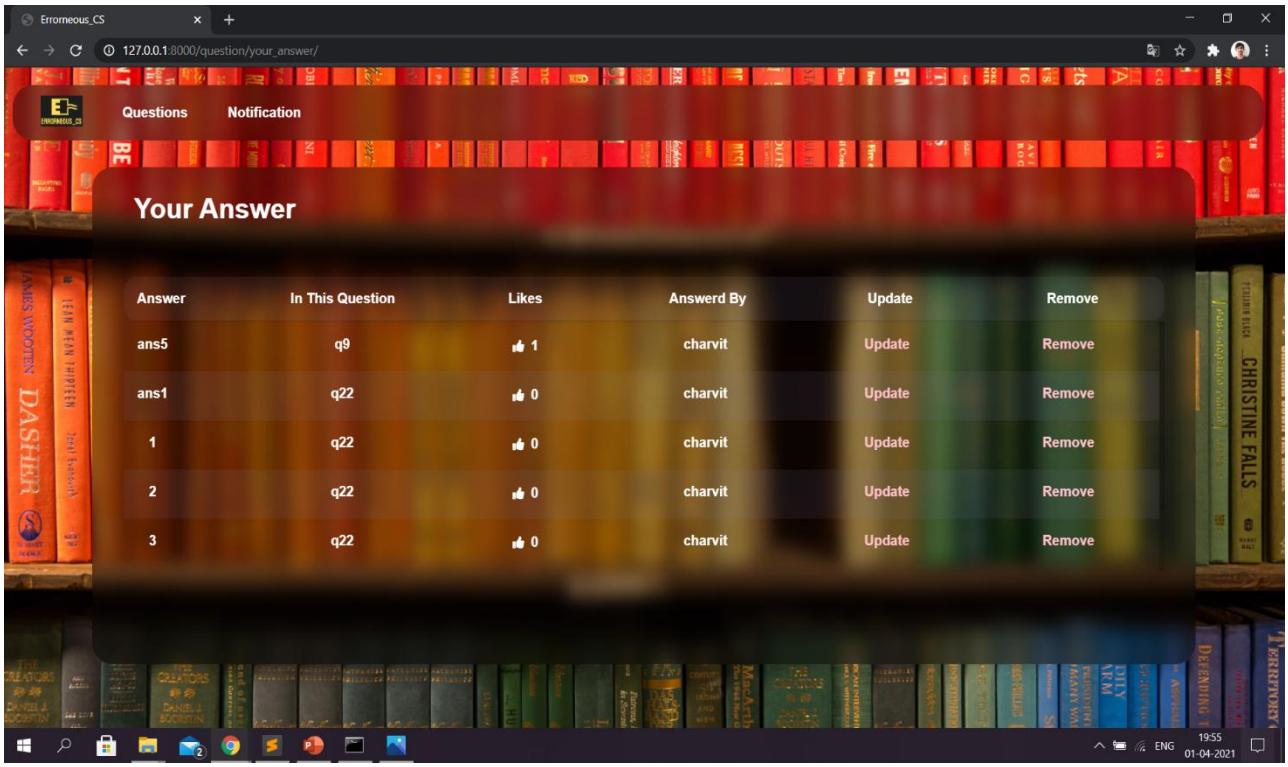
Display Question



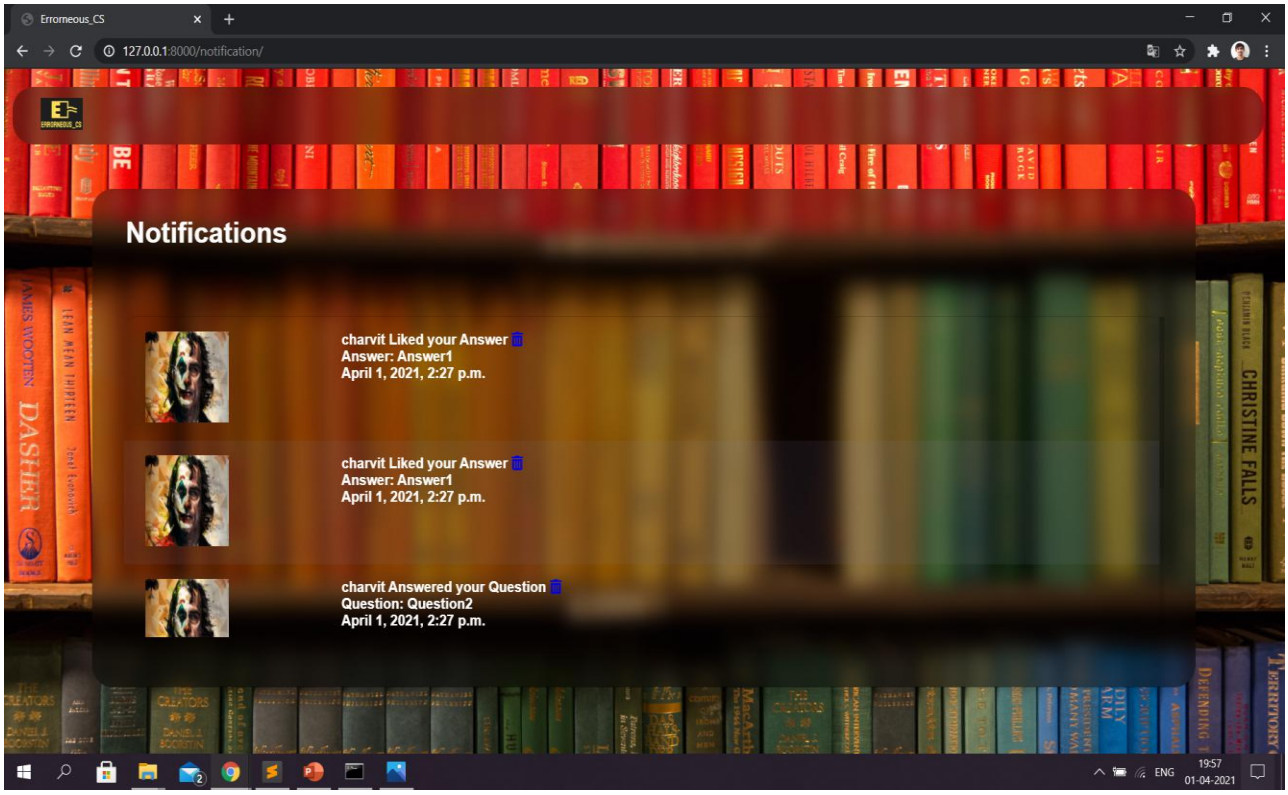
Search Question



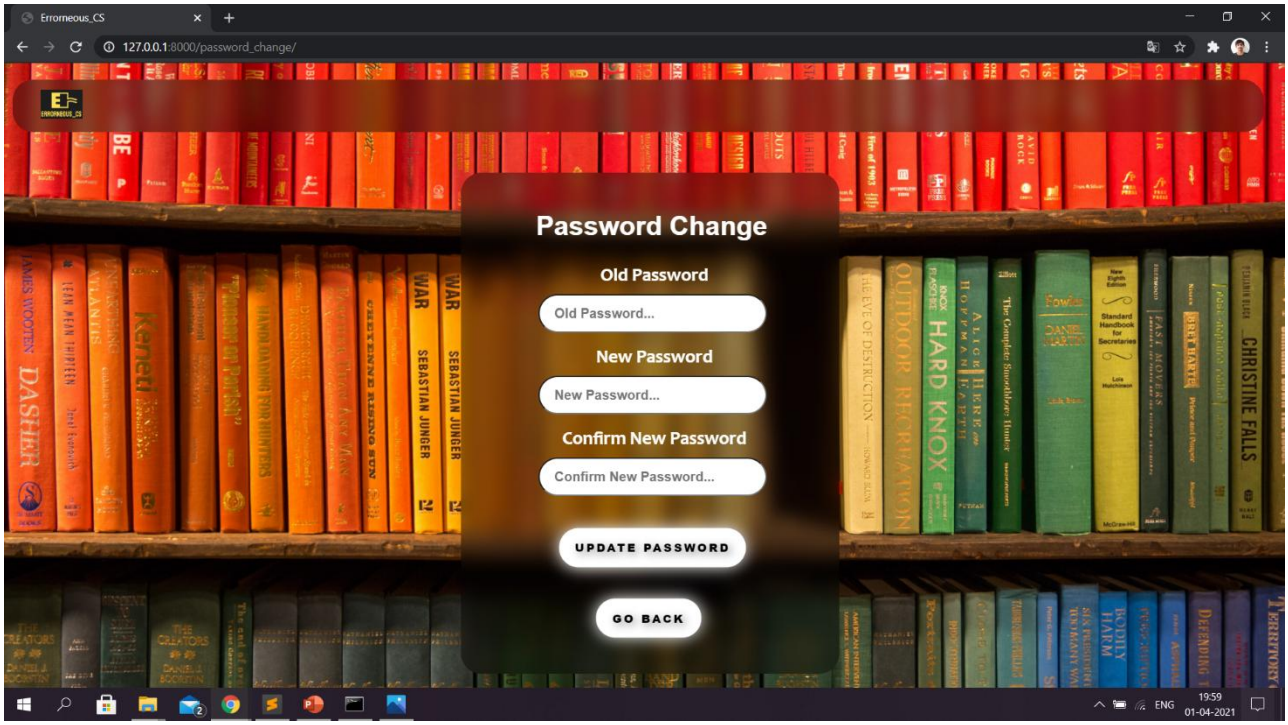
Answer Given By User



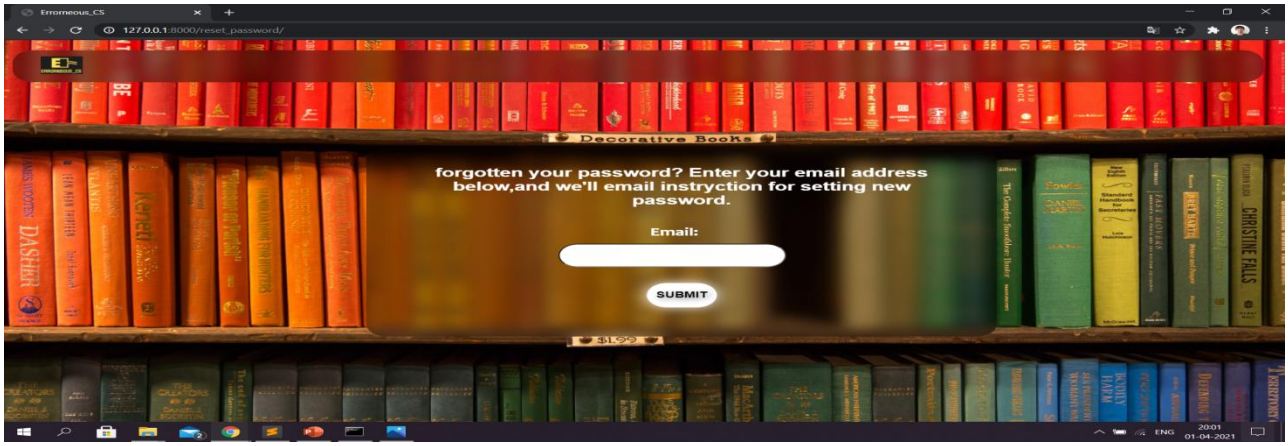
Notification

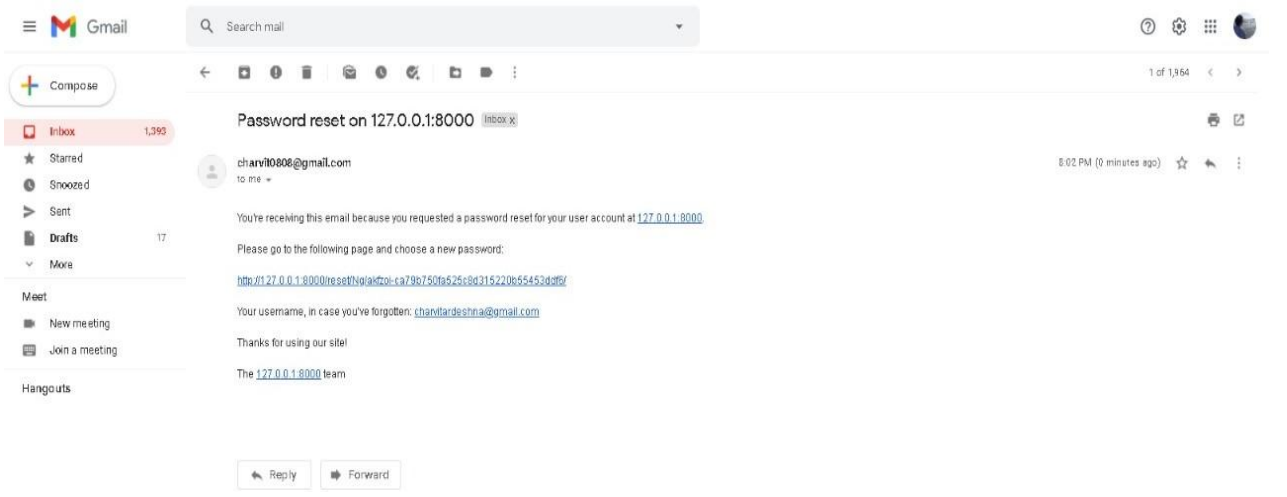


Change Password



Forgot Password





Conclusion

Hence-forth in this project we have successfully implemented the Admin-side & User-side functionality, User can login to the System ,Add Question and Give Answers. According to the system ,it will display Question to Users and User can answer particular Questions. User Can give like to others questions and get notified if user's questions/answers liked by other users. Basically we have try to create clone of stackoverflow.com

Limitations

I. Users can not get notification of dislike.

II. Users can not delete their profile.

III. Users can not upload photo with post.

IV. User can not highlight certain text.

Future Extension

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

I. Add feature of Dislike on Answer and Question

II. chat function

III. User will get feature of tag to question to make search easy.

IV. Add view to question then we will modify page views by sorting the views of questions(add 1 view as any user see that question)

V. We have yet to implement support for mobile tablet views

Bibliography

References/resources used for developing project:

I.docs.djangoproject.com

II. https://stackoverflow.com/

III. https://www.w3schools.com/html/html_css.asp

Iv. https://www.youtube.com/

Git Repository

https://github.com/Charvit123/errorneous_cs.git