**Travel Budgeting and Expense Calculator**

by

Kartik Narang 18BCE1199

A project report submitted to

**Dr.SATHIS KUMAR B.**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

in partial fulfilment of the requirements for the course of

**CSE3001 – SOFTWARE ENGINEERING**

in

**B.Tech. COMPUTER SCIENCE AND ENGINEERING**



**VIT CHENNAI**

**Vandalur – Kelambakkam Road**

**Chennai – 600127**

# NOVEMBER 2019

|  |  |  |
| --- | --- | --- |
| **S no.** | **Description** | **Page no.** |
| 1 | *Bonafide Certificate* | 3 |
| 2 | *Abstract* | 4 |
| 3 | *Acknowledgements* | 5 |
| 4 | *Problem Statement* | 6 |
| 5 | *Functional Requirements* | 8 |
| 6 | *Non-Functional Requirements* | 8 |
| 7 | *Use Case Diagram* | 9 |
| 8 | *Activity Diagram* | 10 |
| 9 | *Sequence Diagram* | 11 |
| 10 | *Task Analysis Chart* | 16 |
| 11 | *Use Case Estimation* | 17 |
| 12 | *Class Diagram* | 18 |
| 13 | *Deployment Diagram* | 19 |
| 14 | *Sample Code* | 20 |
| 15 | *Use Interface Design* | 25 |
| 16 | *Black Box Test Case* | 26 |
| 17 | *White Box Test Case* | 27 |
| 18 | *Conclusion* | 27 |
| 19 | *Future Work* | 28 |

**INDEX**

## BONAFIDE CERTIFICATE

Certified that this project report entitled “**Travel Budgeting and Expense Calculator”** is a bonafide work of **Kartik Narang(18BCE1199)** who carried out the Project work under my supervision and guidance.

**Dr. SATHIS KUMAR B**

Associate Professor

School of Computing Science and Engineering (SCSE), VIT University, Chennai

Chennai – 600127.

## 2. ABSTRACT

The objective of this project was to create an online portal for holiday booking purpose , which is capable of creating packages, registering users and business owners, accepting payments, solving customer queries and generating total trip cost based on given criterias.

### 3. ACKNOWLEDGEMENT

We are extremely grateful to **Dr.Jagdeesh Kannan R,** Dean of the School of Computer Science and Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Programme Chair **Dr.Justus A**for his support throughout the course of this project.

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr.Sathis Kumar B.** , School of Computing Science and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

I thank my parents, family, and friends for bearing with me throughout the course of this project and for the opportunity they provided me in undergoing this course in such a prestigious institution.

**Kartik Narang**

### 4. PPROBLEM STATEMENT

|  |  |
| --- | --- |
| **Name of the project** | Travel Budgeting and Expense Calculator |
| **Objective** | The main objective of the project will be to develop a tool for the usage of Middle Class Household, which while going on vacations, have a fixed budget and the usual unaccounted-for expenses disturb the holidays. |
| **Users of the system** | * The Upper middle class and the Lower-Upper class families. * Tour and Travel Planners |
| **Functional requirements** | * The System is designed to propose possible destinations on a given budget. * The system should keep track of miscellaneous Expenses based on user input and database records. * Database should keep record of different activities and their expenses. * The database should have access to the travelling options like Trains, Buses, Cabs and personal cars. * The system should be able to calculate travelling Expenses based on distance and Vehicle type. * The usual expenses of food for different types of budget demands, their locations, timings, prices etc. should all be accounted for. * E-Payment facilities should be included within the project. |
| **User Inter face** | * There are to be three types of logins. * Admin * Users * Business Owners * HotelsRestrauntsBus services * Cab services * Based on the user location and given budget input, the system should pick feasible locations and plan a tour for given number of days. * Business owners should be able to add, delete and modify any services unless not booked already by a customer. |
| **Extension plans** | 1. The company can start their trademark tour plans on regular or special order basis. 2. Occasional camps and off the map destinations can be targeted to increase the economy and give the customers a wide range of experiences. |
| **Non-Functional Requirements** | * The system should have some access to the railways and airways schedules. * The payment gateway should be secure and safe. * A strong server to run the system is required. |

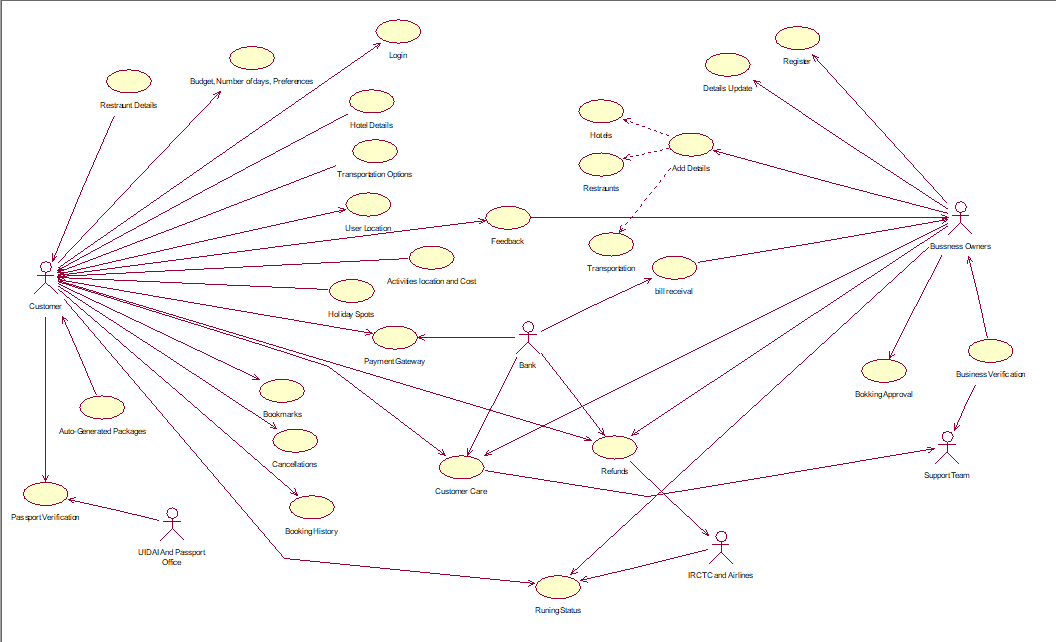
### 5.FUNCTIONAL REQUIREMENTS

1. The System is designed to propose possible destinations on a given budget.
2. The system should keep track of miscellaneous Expenses based on user input and database records.
3. Database should keep record of different activities and their expenses.
4. The database should have access to the travelling options like Trains, Buses, Cabs and personal cars.
5. The system should be able to calculate travelling Expenses based on distance and Vehicle type.
6. The usual expenses of food for different types of budget demands, their locations, timings, prices etc. should all be accounted for.
7. E-Payment facilities should be included within the project.

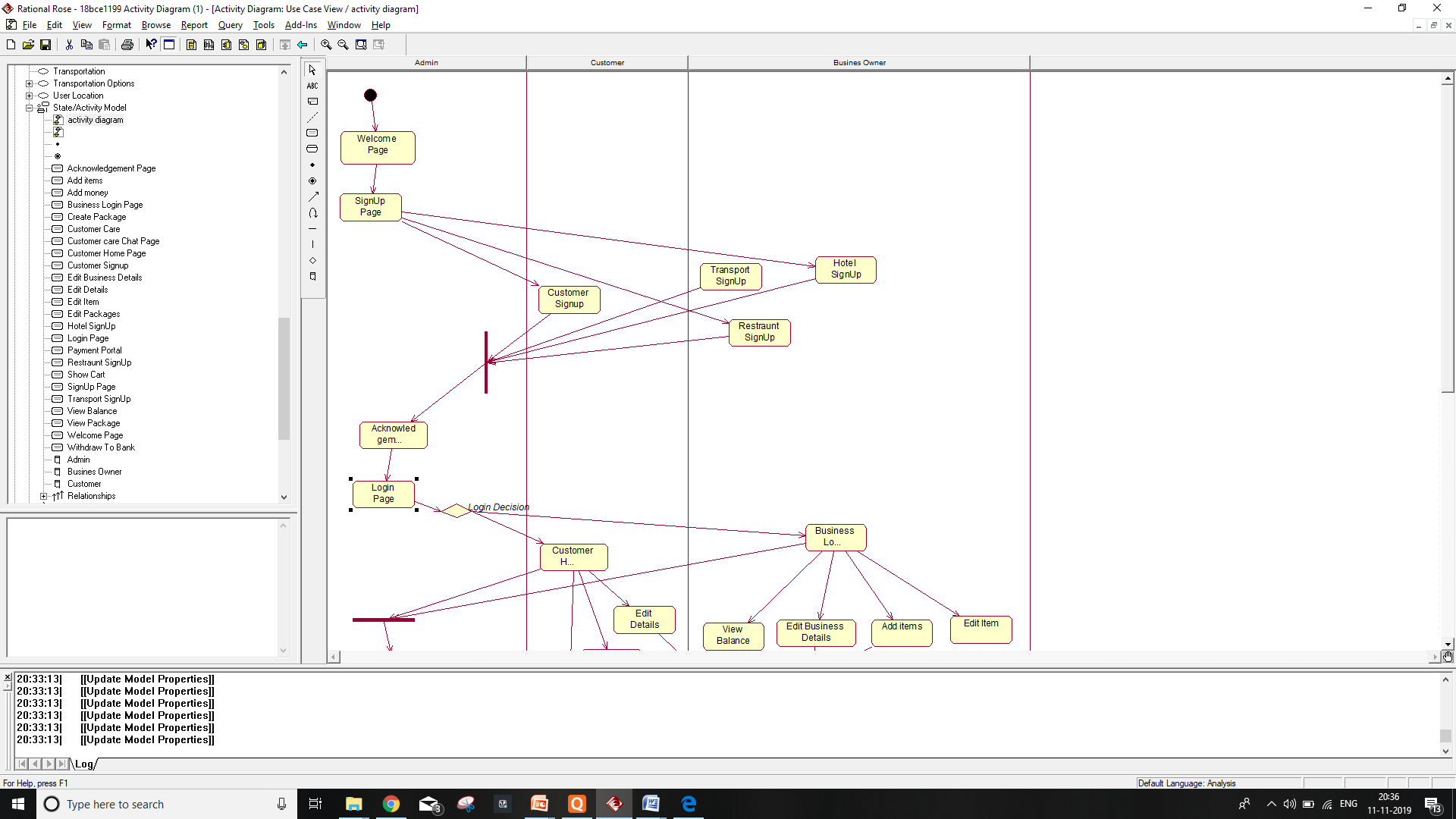
### 6. NON FUNCTIONAL REQUIREMENTS

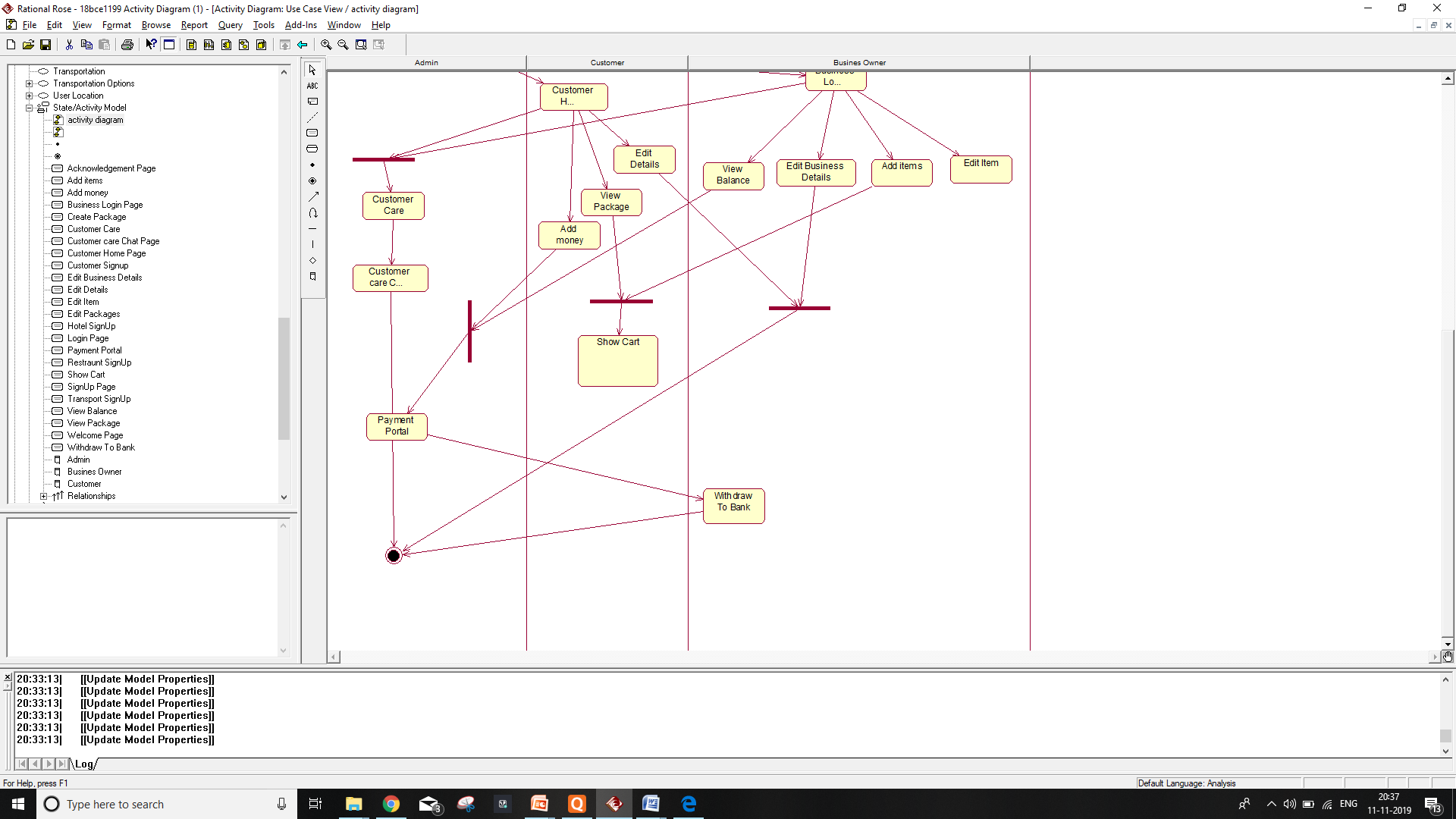
1. The system should have some access to the railways and airways schedules.
2. The payment gateway should be secure and safe.
3. A strong server to run the system is required.

### 7. USE CASE DIAGRAM

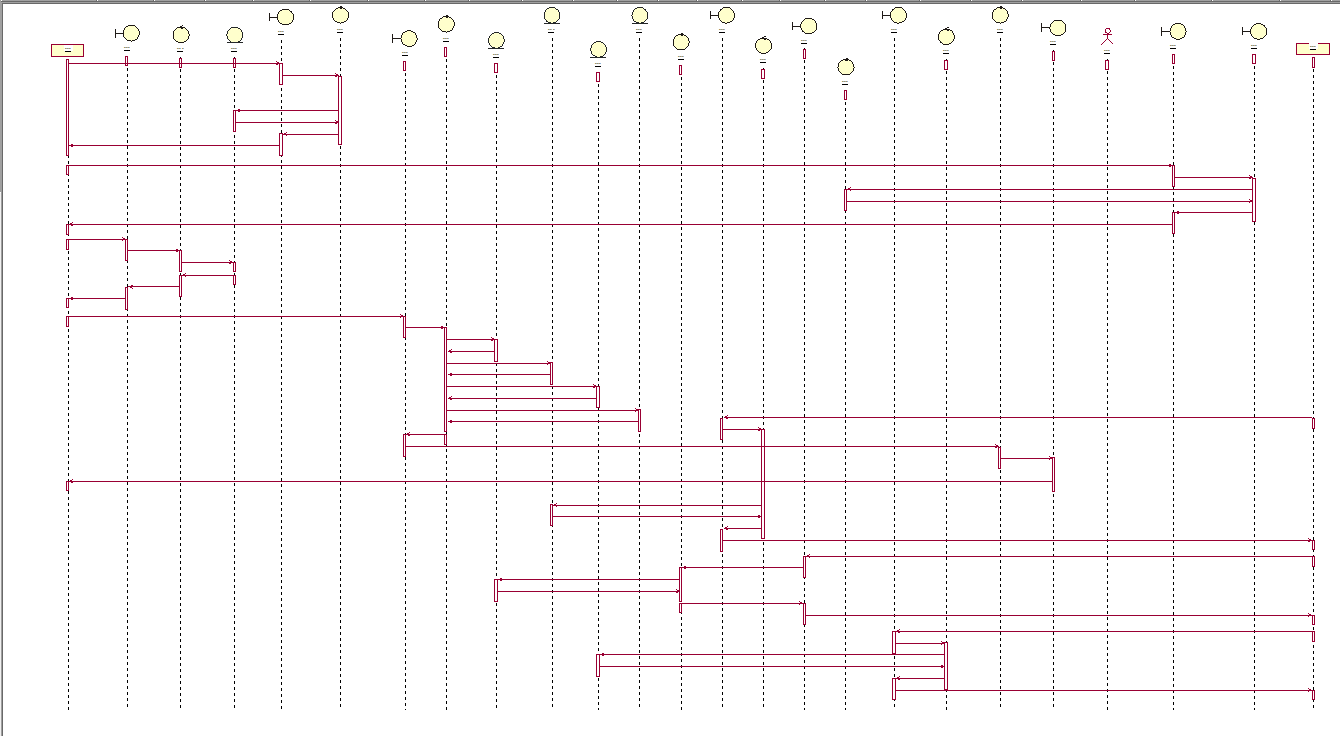
****

**8. ACTIVITY DIAGRAM**

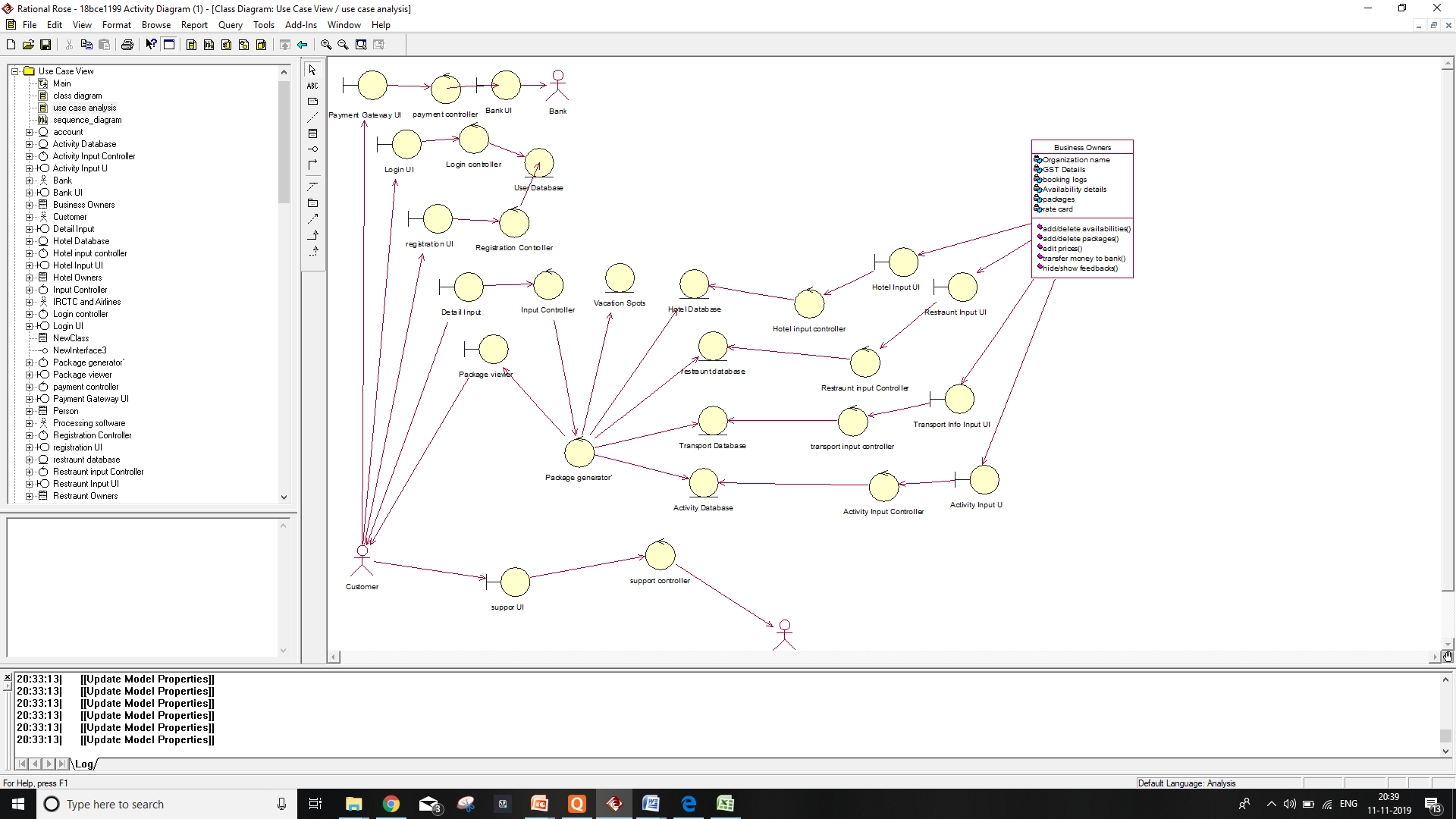




### 9.SEQUENCE DIAGRAM



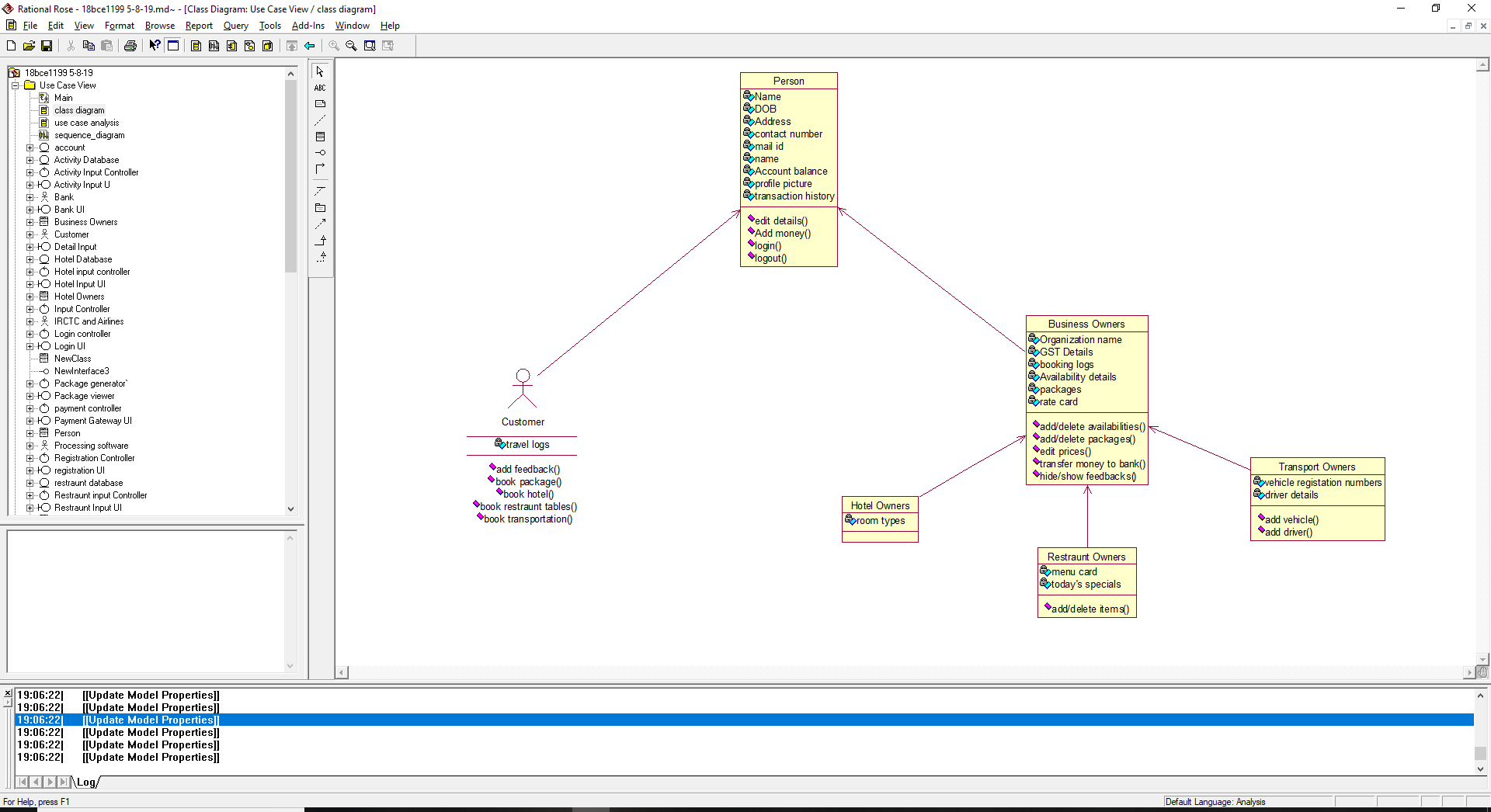
### 10.TASK ANALYSIS CHART



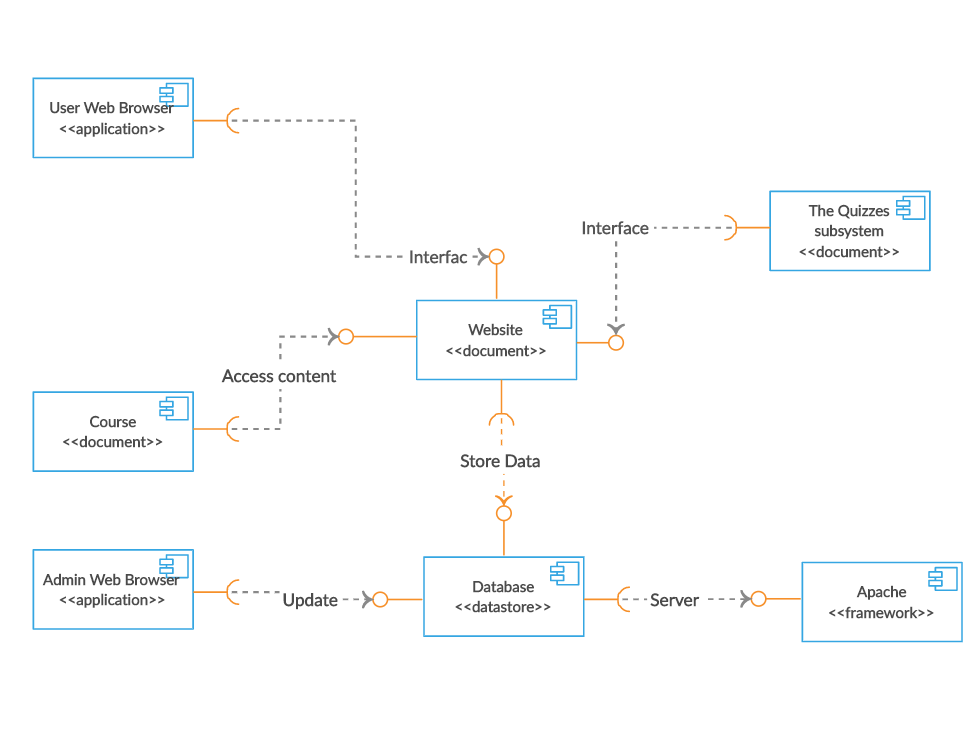
### 11.USE CASE ESTIMATION

|  |  |  |
| --- | --- | --- |
| Calculations From Other Tabs | | |
| TCF | Technical Complexity Factor | 1.03 |
| EF | Environmental Factor | 0.71 |
| UUCP | Unadjusted Use Case Points | 140 |
| AW | Actor Weighting | 8 |
| Calculation of Use Case Points | | |
| UCP | Use Case Points | 108.2 |
| Calculation of Estimated Effort | | |
| Ratio | Hours of Effort per Use Case Point | 28 |
|  | | |
| **Hours of Effort** | | **3,031** |

#### 12. CLASS DIAGRAM



### 13.DEPLOYMENT DIAGRAM



### 14. SAMPLE CODE

from django.db import models

from django.contrib.contenttypes.models import ContentType

from django.contrib.contenttypes.fields import GenericForeignKey

from django.contrib.auth.models import User

from .fields import OrderField

from django.template.loader import render\_to\_string

from django.utils.safestring import mark\_safe

class Subject(models.Model):

    title = models.CharField(max\_length=200)

    slug = models.SlugField(max\_length=200, unique=True)

    class Meta:

        ordering = ('title',)

    def \_\_str\_\_(self):

        return self.title

class Course(models.Model):

    owner = models.ForeignKey(User, on\_delete = models.CASCADE, related\_name='courses\_created')

    subject = models.ForeignKey(Subject,on\_delete = models.CASCADE, related\_name='courses')

    title = models.CharField(max\_length=200)

    slug = models.SlugField(max\_length=200, unique=True)

    overview = models.TextField()

    created = models.DateTimeField(auto\_now\_add=True)

    students = models.ManyToManyField(User,

                                      related\_name='courses\_enrolled',

                                      blank=True)

    class Meta:

        ordering = ('-created',)

    def \_\_str\_\_(self):

        return self.title

class Module(models.Model):

    course = models.ForeignKey(Course,on\_delete = models.CASCADE, related\_name='modules')

    title = models.CharField(max\_length=200)

    description = models.TextField(blank=True)

    order = OrderField(blank=True, for\_fields=['course'])

    class Meta:

        ordering = ['order']

    def \_\_str\_\_(self):

        return '{}. {}'.format(self.order, self.title)

class Content(models.Model):

    module = models.ForeignKey(Module,on\_delete = models.CASCADE, related\_name='contents')

    order = OrderField(blank=True, for\_fields=['module'])

    content\_type = models.ForeignKey(ContentType,on\_delete = models.CASCADE,

                                     limit\_choices\_to={'model\_\_in':('text',

                                                                    'video',

                                                                    'image',

                                                                    'file')})

    object\_id = models.PositiveIntegerField()

    item = GenericForeignKey('content\_type', 'object\_id')

    class Meta:

        ordering = ['order']

class ItemBase(models.Model):

    owner = models.ForeignKey(User,on\_delete = models.CASCADE, related\_name='%(class)s\_related')

    title = models.CharField(max\_length=250)

    created = models.DateTimeField(auto\_now\_add=True)

    updated = models.DateTimeField(auto\_now=True)

    class Meta:

        abstract = True

    def \_\_str\_\_(self):

        return self.title

    def render(self):

        return render\_to\_string('courses/content/{}.html'.format(self.\_meta.model\_name), {'item': self})

class Text(ItemBase):

    content = models.TextField()

class File(ItemBase):

    file = models.FileField(upload\_to='files')

class Image(ItemBase):

    file = models.FileField(upload\_to='images')

class Video(ItemBase):

    url = models.URLField()

from django.urls import reverse\_lazy,reverse

from django.views.generic.edit import CreateView

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth import authenticate, login, logout

from django.views.generic.edit import FormView

from django.views.generic.list import ListView

from django.views.generic.detail import DetailView

from braces.views import LoginRequiredMixin

from courses.models import Course

from django.shortcuts import render

from django.http import HttpResponseRedirect, HttpResponse

from .forms import CourseEnrollForm

from django.contrib.auth.decorators import login\_required

class StudentRegistrationView(CreateView):

    template\_name = 'students/student/registration.html'

    form\_class = UserCreationForm

    success\_url = reverse\_lazy('student\_course\_list')

    def form\_valid(self, form):

        result = super(StudentRegistrationView, self).form\_valid(form)

        cd = form.cleaned\_data

        user = authenticate(username=cd['username'],

                            password=cd['password1'])

        login(self.request, user)

        return result

class StudentEnrollCourseView(LoginRequiredMixin, FormView):

    course = None

    form\_class = CourseEnrollForm

    def form\_valid(self, form):

        self.course = form.cleaned\_data['course']

        self.course.students.add(self.request.user)

        return super(StudentEnrollCourseView, self).form\_valid(form)

    def get\_success\_url(self):

        return reverse\_lazy('student\_course\_detail', args=[self.course.id])

class StudentCourseListView(LoginRequiredMixin, ListView):

    model = Course

    template\_name = 'students/course/list.html'

    def get\_queryset(self):

        qs = super(StudentCourseListView, self).get\_queryset()

        return qs.filter(students\_\_in=[self.request.user])

class StudentCourseDetailView(DetailView):

    model = Course

    template\_name = 'students/course/detail.html'

    def get\_queryset(self):

        qs = super(StudentCourseDetailView, self).get\_queryset()

        return qs.filter(students\_\_in=[self.request.user])

    def get\_context\_data(self, \*\*kwargs):

        context = super(StudentCourseDetailView, self).get\_context\_data(\*\*kwargs)

        # get course object

        course = self.get\_object()

        if 'module\_id' in self.kwargs:

            # get current module

            context['module'] = course.modules.get(id=self.kwargs['module\_id'])

        else:

            # get first module

            context['module'] = course.modules.all()[0]

        return context

class StudentModuleDetailView(DetailView):

    model = Course

    template\_name = 'students/course/course\_detail.html'

    def get\_queryset(self):

        qs = super(StudentModuleDetailView, self).get\_queryset()

        return qs.filter(students\_\_in=[self.request.user])

    def get\_context\_data(self, \*\*kwargs):

        context = super(StudentModuleDetailView, self).get\_context\_data(\*\*kwargs)

        # get course object

        course = self.get\_object()

        if 'module\_id' in self.kwargs:

            # get current module

            context['module'] = course.modules.get(id=self.kwargs['module\_id'])

        else:

            # get first module

            context['module'] = course.modules.all()[0]

        return context

@login\_required

def logoutuser(request):

    logout(request)

    return render(request,'registration/logged\_out.html')

def loginuser(request):

    # login(request,user)

    if request.method == 'POST':

        username = request.POST.get('username')

        password = request.POST.get('password')

        user = authenticate(username=username, password=password)

        if user:

            if user.is\_active:

                login(request,user)

                return HttpResponseRedirect(reverse('course\_list'))

            else:

                return HttpResponse("Your account was inactive.")

        else:

            print("Someone tried to login and failed.")

            print("They used username: {} and password: {}".format(username,password))

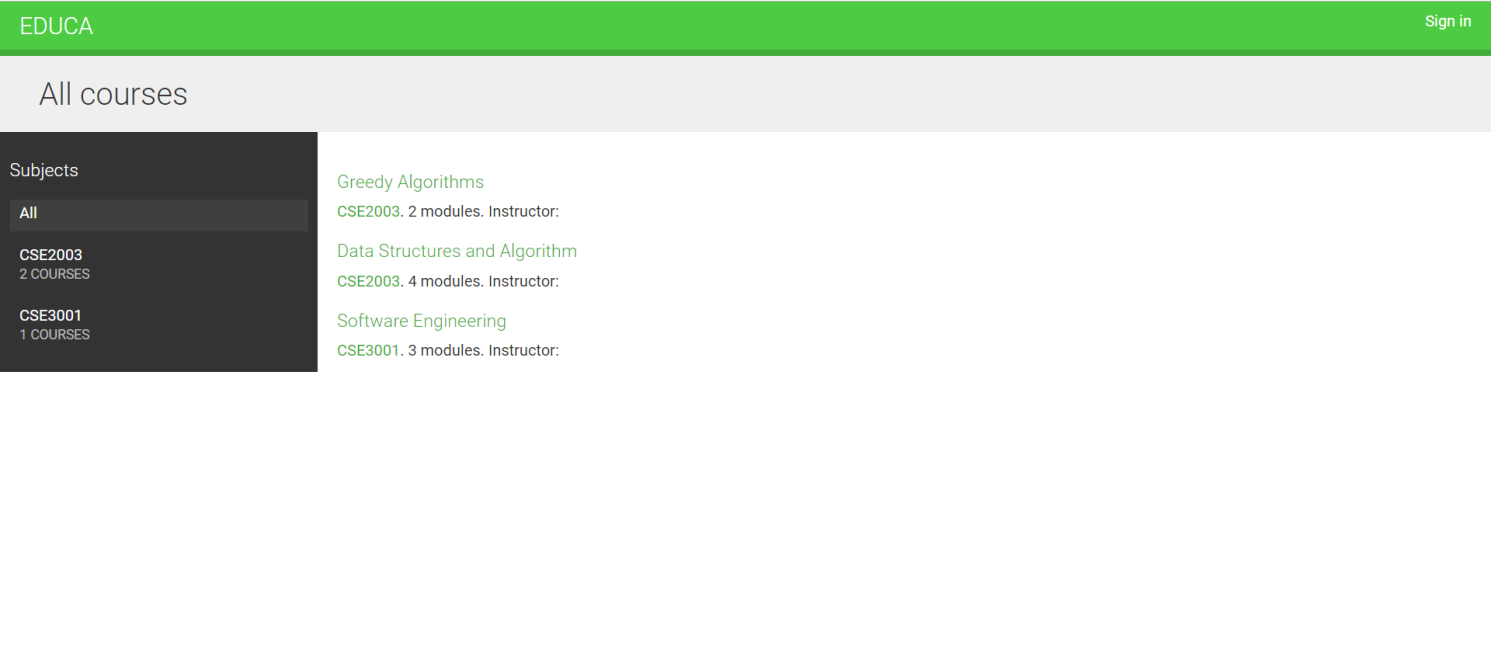
            return HttpResponse("Invalid login details given")

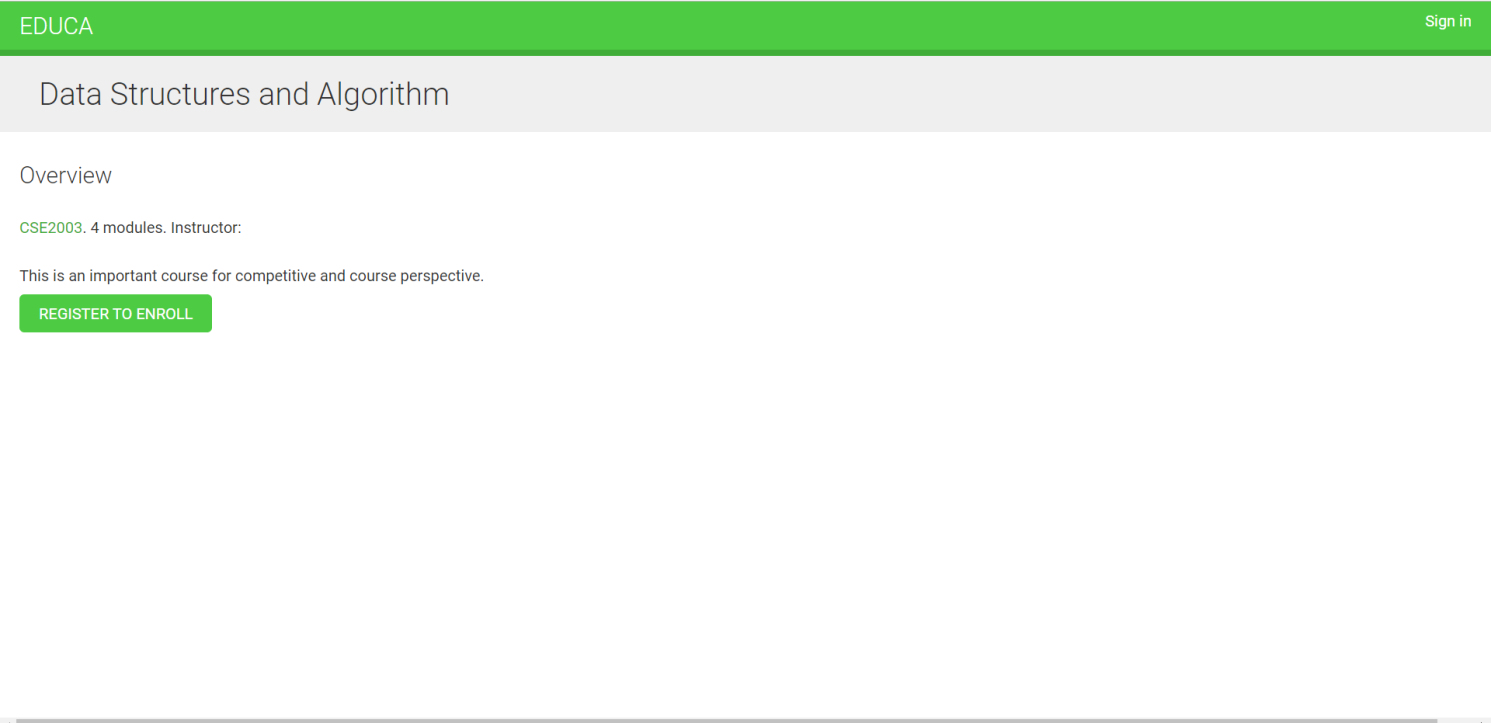
    else:

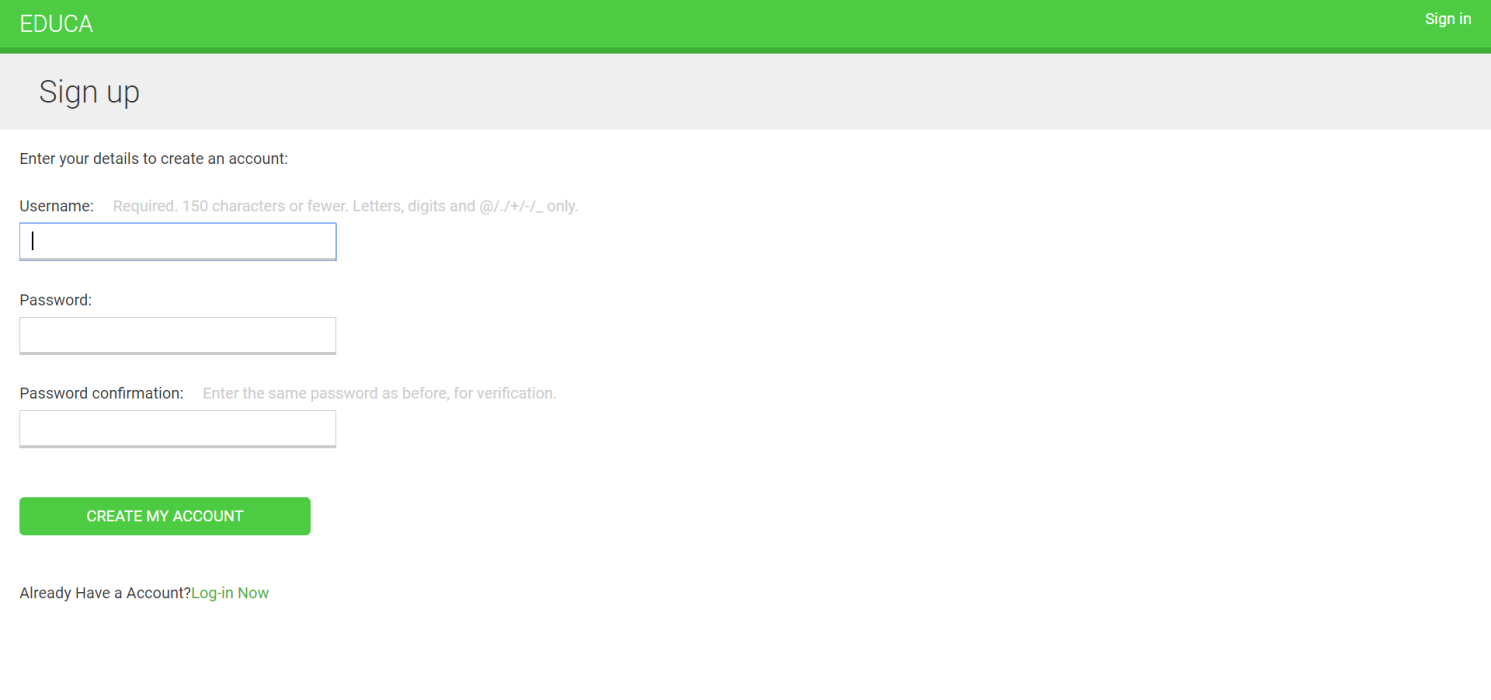
        return render(request, 'registration/login.html', {})

    # return render(request,'registration/login.html')

### 15. USER INTERFACE DESIGN







### 16. BLACK BOX TEST CASE

|  |  |  |
| --- | --- | --- |
| Sr.No | Test Cases | Result |
| 1. | a user will be able to login with a valid username and valid password. | Positive |
| 2. | a user cannot login with a valid username and an invalid password. | Positive |
| 3. | ‘Forgot Password’ functionality. | Negative |
| 4. | data in password field is either visible as asterisk or bullet signs. | Positive |
| 5. | the ‘Enter’ key of the keyboard is working correctly on the login page | Positive |
| 6. | a user cannot enter the characters more than the specified range in each field (Username and Password) | Positive |
| 7. | Login page against SQL injection attack. | Negative |
| 8. | Make a payment without contact details | Negative |
| 9. | Admin has all the rights | Positive |
| 10. | Unregistered users can view the packages | Positive |

### 17. WHITE BOX TEST CASE

|  |  |  |
| --- | --- | --- |
| Sr.No | Test Cases | Result |
| 1. | Models can be updated | Positive |
| 2. | Admin can add or remove users | Positive |
| 3. | Data structures and algorithm course add | Positive |
| 4. | User Kartik Registers and Login . | Positive |
| 5. | Kartik can approve discount codes. | Positive |

### 18. CONCLUSIONS

I successfully implemented the platform for **Holiday Package and expense calculator**.

Basic models for tourists, admin and business owners were successfully implemented using Python and Django framework.

A registered tourist can successfully book a holiday and business owners can be added by the admin or by themselves using signup.

The results received were accepted and verified.

### 19. FUTURE WORK

The Project can be further extended from web portal to mobile application.

And Mobile wallet can be implemented into the project which could help the tourist-business interaction.

Further, the portal can also be made to use for advertisements using external ad services.