Module	Portfolio	Assessment Type
Collaborative Development (5CS024)	1	Individual Report

Tour Management System - Developer

Student Id : 2059624

Student Name : Pallaw Rana Magar

Section : L5CG7

Group : L5CG7 Group 4

Module Leader : Er. Uday Kandel

Lecturer : Er. Raj Prasad Shrestha

Acknowledgement

I wish to express my sincere gratitude to University of Wolverhampton and Herald College for giving me the opportunity to write this report.

I would also like to thank my teacher Mr. Biraj Dulal for guiding me in completing this report. Finally, I would also like to thank my group members for their effort on completing this project.

Contents

Table of Figure

Self-appraisal form

Student number	2059624	Name	Pallaw Rana Magar
Project	Tour Management System	Date	04/11/2022
Role	Backend Developer	Team	L5CG7 (group 4)
Sprint (1 or 2)	1		

Personal objectives – performance measurement

These should be copied from your role description.

Objectives	**Evidence provided** *(E.g. appendix A, file name etc.)*	**Evaluation** *Student / tutor*	
Choosing for relevant technologies	I chose to do this project with Django as it seemed to be the best option. Since its main job is to build and maintain web application. And I chose to work with its default database SQLite for this Sprint as SQLite can support a wide variety of platforms such as Windows, iOS, etc. Appendix A	9	
Tutor feedback:			
Implementing functional requirements	For this sprint, we have created many applications such newsletter, booking, packages for the project. The admin can add new packages in admin panel. And the user can view the packages and can book the packages by filling in their information. Users can also subscribe to newsletter to receive any news regarding the packages. Appendix A	9	
Tutor feedback:			
		/20	/20

Evidence Provided: The reference of the appendix in which the evidence can be found (*Appendix A, appendix B, screenshot.jpg, myreport.docx* etc. please attach to this form). Don't hesitate to add comments and notes to your appendices to highlight particular sections, relevant pieces of code etc.

Collaboration Document

Evidence of good collaboration

Good communication and file sharing

Receivable evidence includes:

- Emails and other types of messages (Facebook messenger, WhatsApp etc.)
- Screenshots of Basecamp conversations in which you actively participate.
- Screenshots showing files (designs, reports) that **you** shared with your team on Basecamp

***Important**: Please include no more than 5 items*

I communicated with my team through different software's such as Facebook, Discord and Basecamp. We would talk informally in Facebook and talk about what to do, how to do. We would also talk in Basecamp and send the work did ask for feedback. I also talked in Discord with project manager and sent him my works to check if it was done correctly. Appendix 2

Continuing Personal Development (CPD)

Receivable evidence includes:

- Course/seminar attendance register
- Online course: certificate of completion
- Word document summarising what was learnt and how it can be used on the project

***Important**: Please include no more than 5 items*

I watched a course posted by freecodecamp.org in YouTube to learn Django. I learned all the basics from that video from how to setup a virtual environment, models, templates, etc which were extremely helpful in my project. Another resource was a book called Django for APIs. I learned about REST Framework and how to make an API with the help of this book. Appendix 2

Issue tracking

Receivable evidence includes:

- Screenshots of **personal contributions** to GitHub issues.

***Important**: Please include no more than 5 items*

In the first issue I had was having less display fields in bookings model, I fixed it by adding more display fields in booking models of admin panel. I fixed the second issue by adding a comment about username and password of the superuser. Then, I fixed the third issue by updating the requirements.txt file as it was showing unused package.

Appendix 3

Work to deadlines

No evidence required. Your tutor will decide whether you have worked to deadlines based on various factors (team meetings, discussions with other team members, discussion with client etc.)

Appendix A

Choosing for relevant technologies

Choosing a new language for a project can be considered a very important step while building a project. Before choosing a new language or framework, we often look for simplicity, reliability, security, built- in packages, versatility, etc. Python is a very popular language which can fulfil all these requirements. Due to the availability of this many features, this language can be considered as one of the best. Django being a Python-based framework is also very popular right now for building web-based applications.

For this project I chose to work with Django as it was the best option. There are many popular languages such as PHP, Laravel, etc. All these languages have their pros and cons. Django is a framework, and PHP is a web development language. Django's main job is to build and maintain web applications whereas PHP allows to create dynamic content which helps in interacting with database. Django provides built-in support for many databases such as PostgreSQL, MariaDB, MySQL, Oracle, SQLite, etc. whereas in PHP, we don't have the facility to choose between databases. PHP supports primary databases like MySQL, Oracle, etc. It also offers a PDO layer that constantly interacts with our app and databases services. Django works flawlessly with various technologies while maintaining optimal performance. Django also does a great job while optimizing elements like DB, images, CSS, etc. and balancing the load between resources which can be very helpful in the project (Dhaduk, 2021). While comparing security standards, PHP offers secure websites but needs skilful and experienced developers whereas Django covers all the security loopholes of PHP. File uploading is also very simpler in Django than PHP. In Django, we can use the Image field while Django takes care of validating the images. But in PHP, we need to write a bunch of code and PHP also won't be checking for any errors (Data flair, n.d.).

And for this project I used the default database of Django which is SQLite. I might be changing the database in the next Sprint to MySQL or some other database. One of the main advantages of using SQLite is that it has a wide variety of platforms like Windows, MacOS, iOS, etc. SQLite database doesn't need any additional server

process as it is implemented on a single file. SQLite also doesn’t need a server to run while MySQL requires a database server to run. (Duggal, 2021)

While Django and SQLite may have these advantages, we also need to understand their limitations. For Django, it is not applicable for smaller projects. It is an intensive framework which takes a lot of bandwidth and processing time. Django also cannot handle multiple requests at a time like other frameworks. It also has slow evolution which means modules needs to be developed with backward compatibility (Dhaduk, 2021). For SQLite, it has limited size which means it can only support database up to 140 TB. It also doesn’t provide direct access to data rather the access is built into the system. Also, we can only write one application at a time (Duggal, 2021).

Implementing functional requirement

```python
from django.db import models
from django.contrib.auth.models import User
from django.utils.text import slugify
from ckeditor.fields import RichTextField
from PIL import Image
from io import BytesIO
from django.core.files.uploadedfile import InMemoryUploadedFile as imuf

# Created database to store tour package
# Create your models here.
class TourPackage(models.Model):
    title = models.CharField(max_length=50)
    slug = models.SlugField(max_length=50, blank=True)
    image = models.ImageField(blank = True, upload_to='tourpackage_images')
    thumbnail_Image = models.ImageField(blank = True, upload_to = 'tourpackage_images')
    no_of_days = models.IntegerField()
    summary = models.TextField(blank= True)
    full_detail = RichTextField(blank = True)
    is_featured = models.BooleanField(default=False)
    is_active = models.BooleanField()
    price = models.IntegerField()

    def __str__(self):
        return self.title
# representing object by its title

    def save(self, *args, **kwargs):
        if (not self.slug):
            self.slug = slugify(self.title)
        self.thumbnail_Image = self.compressImage(self.image)
```

Figure 1 Screenshot of models.py

```python
# representing object by its title

    def save(self, *args, **kwargs):
        if (not self.slug):
            self.slug = slugify(self.title)
        self.thumbnail_Image = self.compressImage(self.image)
        super(TourPackage, self).save(*args, **kwargs)

    def compressImage(self, full_image):
        thumb_image = Image.open(full_image)
        # Convert to RGB as RGBA can not be changed into JPEG(no transparency)
        thumb_image = thumb_image.convert('RGB')
        fName = full_image.name.split('.')[0]
        outio = BytesIO()
        # Save the new file
        thumb_image.save(outio, format='JPEG', quality=30)
        uImage = imuf(outio, 'ImageField', f"{fName}_thumbnail.jpg", 'image/jpeg', outio.tell(), None)
        # return that file name to store in Database
        return uImage
```

Figure 2 Screenshot of models.py

This is the models.py of tourPackage I made. In the first class TourPackage(), here I have fields. This is used to create database to store tour packages with objects we

have provided. Then, in function save and compress Image, we use it to create a thumbnail image by compressing it.

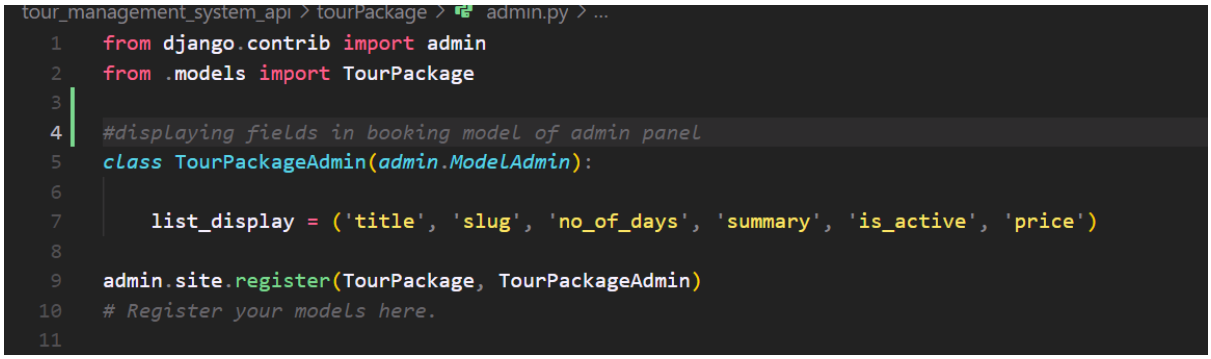

Figure 3 Screenshot of admin.py

In this screenshot I have created a class where we have different fields which will be displayed in bookings of admin panel.

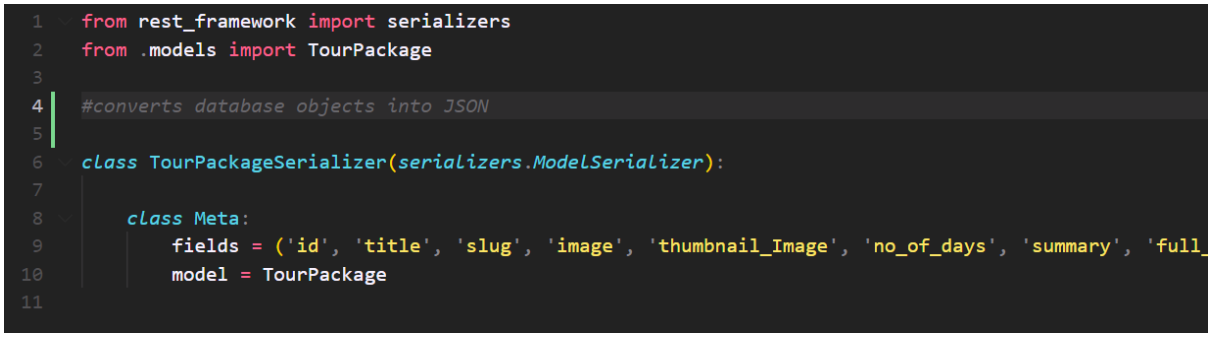

Figure 4 Screenshot of serializers.py

This screenshot is of the app serializers.py which coverts the database objects from data in JSON.

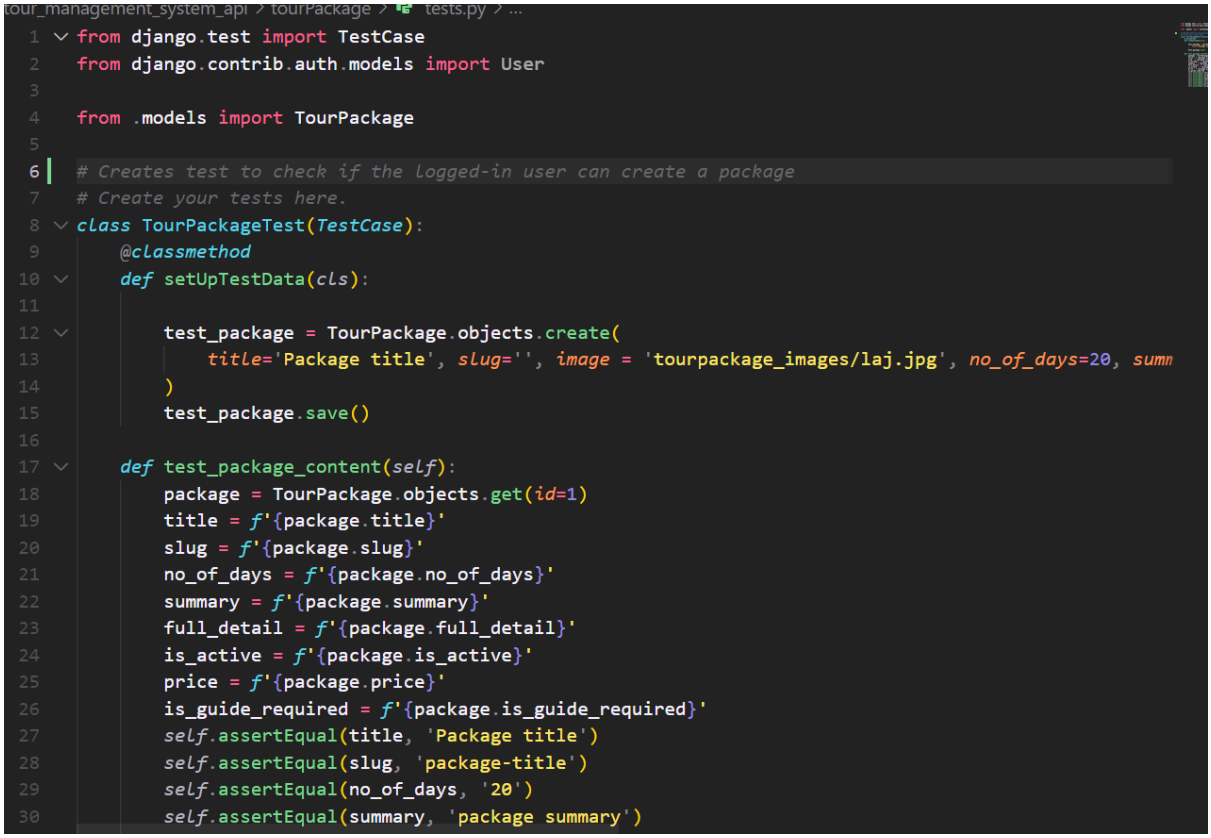

Figure 5 Screenshot of tests.py

This screenshot is of tests.py which we use to create test to check if the logged-in user can create a package as expected.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    #3rd party apps
    'rest_framework', #new
    'ckeditor',
    'corsheaders',
    #Local
    'tourPackage.apps.TourpackageConfig',
    'newsletter.apps.NewsletterConfig',
    'bookings.apps.BookingsConfig',
]

REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.AllowAny',
    ]
}
```

Figure 6 Screenshot of settings.py

This is settings.py where we add our installed packages whether it is a 3rd party app or local application.

```python
from django.contrib import admin
from django.urls import include, path
from django.conf import settings
from django.conf.urls.static import static

#urls of API
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/tour-packages/', include('tourPackage.urls')),
    path('api/bookings/', include('bookings.urls')),
    path('api/newsletter/', include('newsletter.urls')),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Figure 7 Screenshot of urls.py

This screenshot shows the URLs for the APIs I created.

Use of Version Control

This is the screenshot which shows my git log.

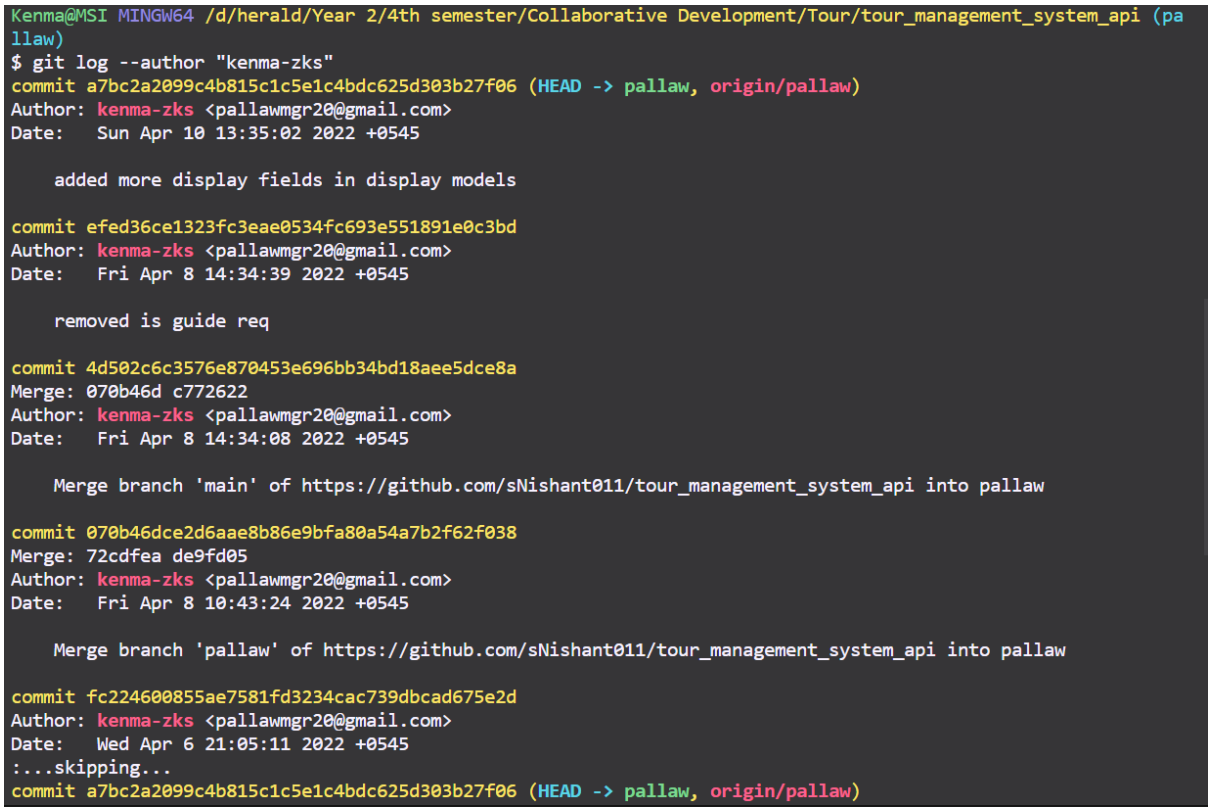

Figure 8 Screenshot of git log

```
$ git log --graph
* commit a7bc2a2099c4b815c1c5e1c4bdc625d303b27f06 (HEAD -> pallaw, origin/pallaw)
| Author: kenma-zks <pallawmgr20@gmail.com>
| Date:   Sun Apr 10 13:35:02 2022 +0545
|
|     added more display fields in display models
|
*   commit c07e959866e7faf97f797fd85142981b68c36ba2 (origin/main, origin/HEAD)
|\  Merge: c772622 efed36c
| | Author: Nishant Shrestha <49269819+sNishant011@users.noreply.github.com>
| | Date:   Fri Apr 8 14:37:09 2022 +0545
| |
| |     Merge pull request #12 from sNishant011/pallaw
| |
| |     added is_featured in package model
| |
| * commit efed36ce1323fc3eae0534fc693e551891e0c3bd
| | Author: kenma-zks <pallawmgr20@gmail.com>
| | Date:   Fri Apr 8 14:34:39 2022 +0545
| |
| |     removed is guide req
| |
| *   commit 4d502c6c3576e870453e696bb34bd18aee5dce8a
| |\  Merge: 070b46d c772622
| |/  Author: kenma-zks <pallawmgr20@gmail.com>
|/|   Date:   Fri Apr 8 14:34:08 2022 +0545
| |
| |       Merge branch 'main' of https://github.com/sNishant011/tour_management_system_api into pallaw
| |
* |   commit c772622290e0566b6c95767c0abfc7775b9f4189
```

```
* commit a7bc2a2099c4b815c1c5e1c4bdc625d303b27f06 (HEAD -> pallaw, origin/pallaw)
| Author: kenma-zks <pallawmgr20@gmail.com>
| Date:   Sun Apr 10 13:35:02 2022 +0545
|
|     added more display fields in display models
|
*   commit c07e959866e7faf97f797fd85142981b68c36ba2 (origin/main, origin/HEAD)
|\  Merge: c772622 efed36c
| | Author: Nishant Shrestha <49269819+sNishant011@users.noreply.github.com>
| | Date:   Fri Apr 8 14:37:09 2022 +0545
| |
| |     Merge pull request #12 from sNishant011/pallaw
| |
| |     added is_featured in package model
| |
| * commit efed36ce1323fc3eae0534fc693e551891e0c3bd
| | Author: kenma-zks <pallawmgr20@gmail.com>
| | Date:   Fri Apr 8 14:34:39 2022 +0545
| |
| |     removed is guide req
| |
| *   commit 4d502c6c3576e870453e696bb34bd18aee5dce8a
| |\  Merge: 070b46d c772622
| |/  Author: kenma-zks <pallawmgr20@gmail.com>
|/|   Date:   Fri Apr 8 14:34:08 2022 +0545
| |
| |       Merge branch 'main' of https://github.com/sNishant011/tour_management_system_api into pallaw
| |
* |   commit c772622290e0566b6c95767c0abfc7775b9f4189
|\ \  Merge: 72cdfea de9fd05
| | | Author: Nishant Shrestha <49269819+sNishant011@users.noreply.github.com>
| | | Date:   Fri Apr 8 09:08:31 2022 +0545
| | |
| | |     Merge pull request #11 from sNishant011/pallaw
| | |
| | |     changed some field names and serializer
```

Figure 9 Screenshot of git graph

This is the git graph done for the project.

Appendix 2

Evidence of Good communication and file sharing

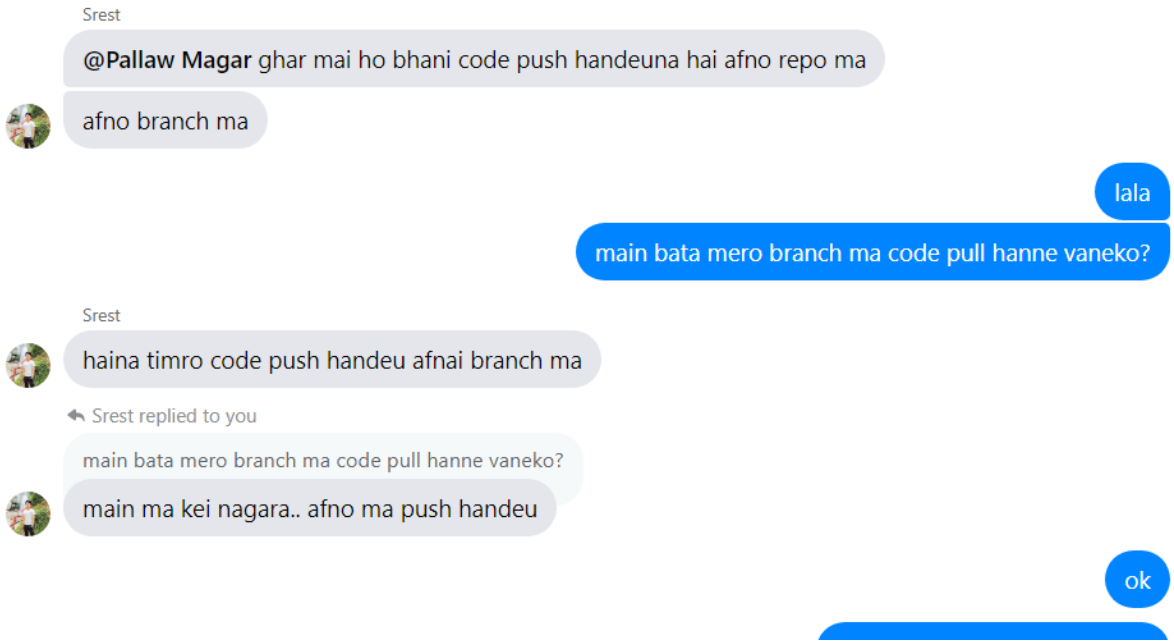

Figure 10 Messages in Facebook

Screenshots of members chatting in messenger.

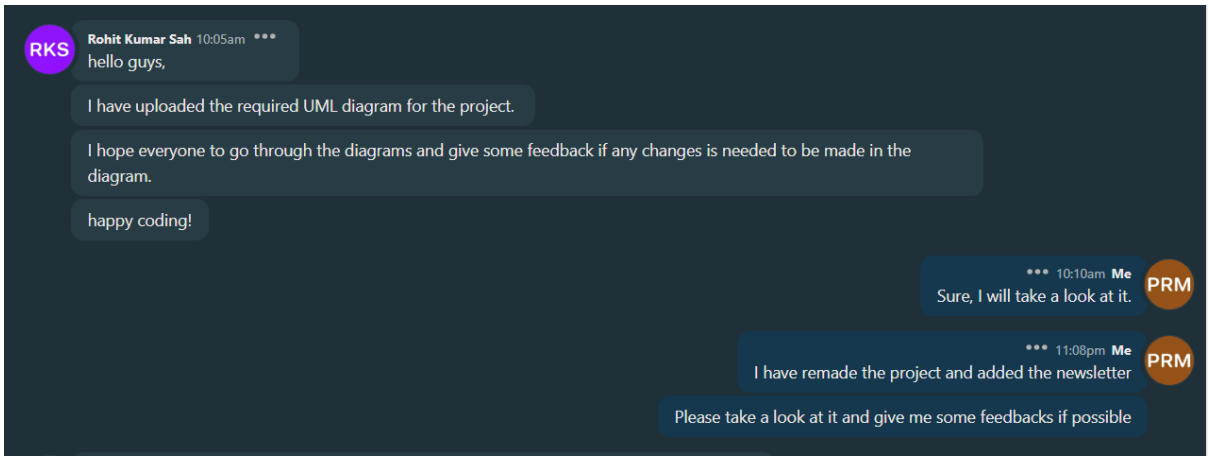

Figure 11 Communicating in Basecamp

Screenshot of team chatting in Campfire.

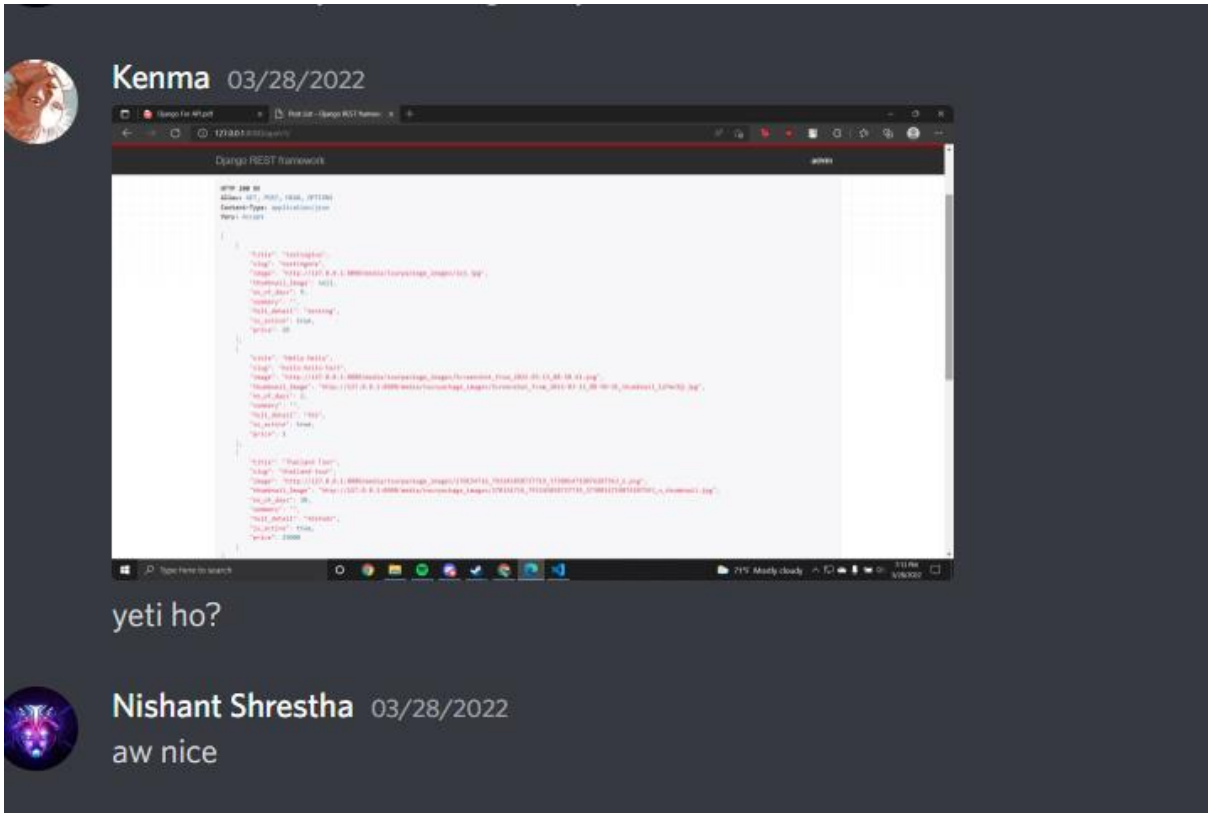

Figure 12 Sending resources in discord

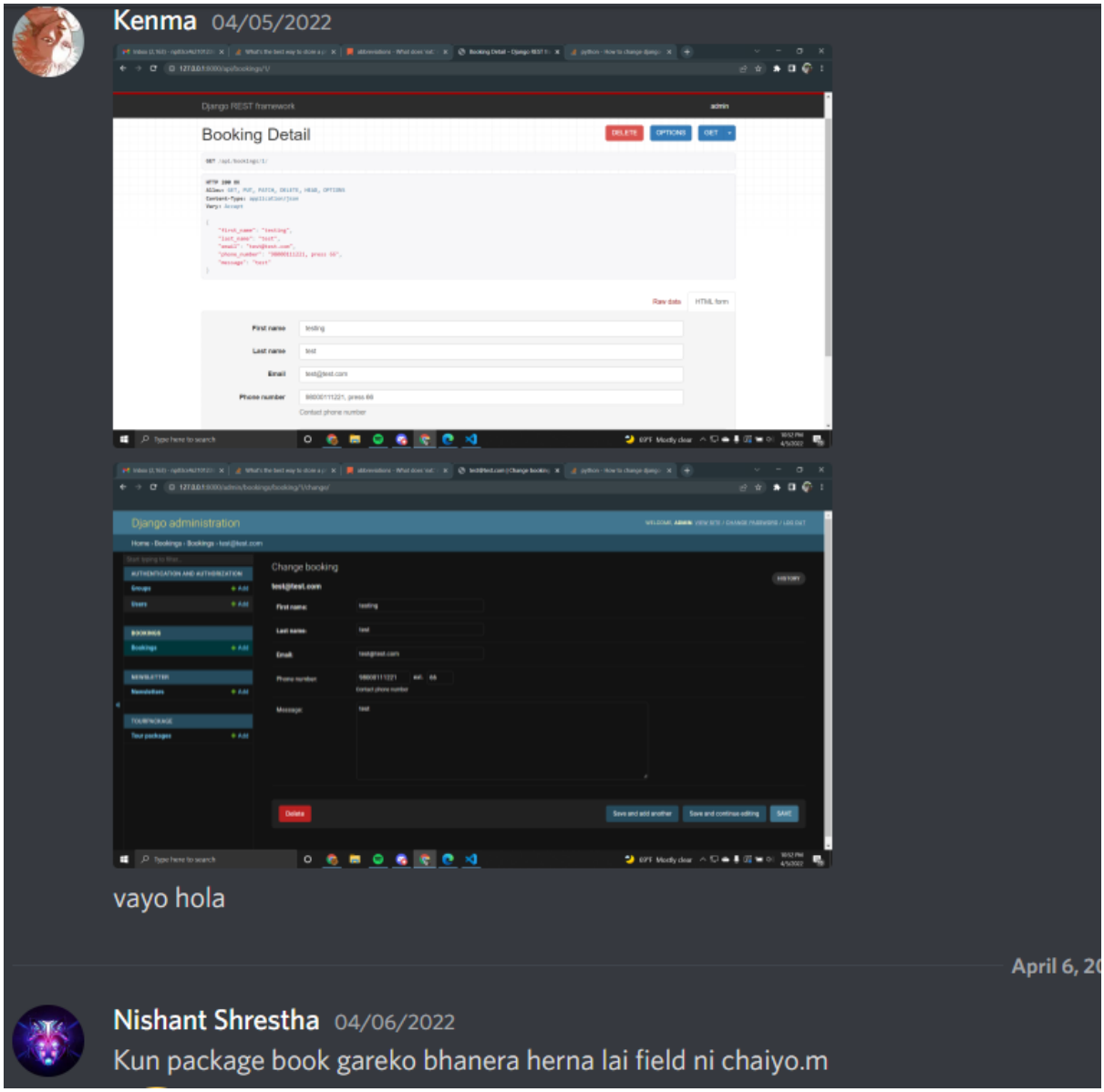

Figure 13 Sending resources in Discord

Screenshot of me sending resources on Discord to our Project Manager requesting for feedback to check if the work is done correctly,

Evidence of Continuing Personal Development (CPD)

I had to learn Django to do this project. I had no previous knowledge of Django, so I had to learn it from scratch. I learned Django by watching the video posted by freeCodeCamp.org in YouTube. They had a full course for Beginners about Django Web Framework. In that video, Django was taught from the basics from how to setup environment to Class based views. I learned how to setup a virtual environment and learned about different fields available in models.py. I also learned how to change a model and learned about URL routings, templates. In the video, every step was explained deeply such as how to render context in templates, using conditions in templates, etc. The video also had information regarding model forms. All these topics helped me greatly in starting Django and helped me in building my first project. I used different fields used in models.py, I was able to setup a virtual environment thanks to this video. I also used URL routings in my project to route into my API.

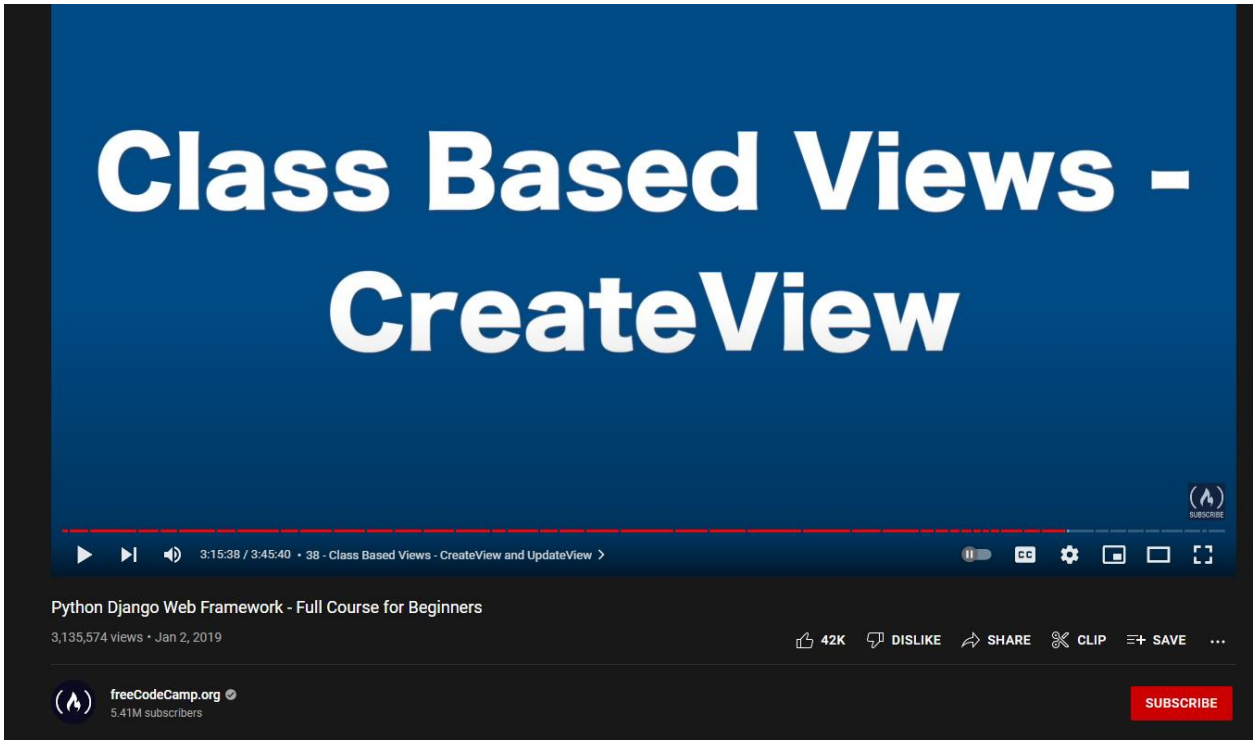

Figure 14 Screenshot of course

Here is the picture of the course that helped me in the beginning to start the Django project and how to work with Django.

Another resource which helped me greatly in the project was Django for API written by William S. Vincent. This book helped in building multiple RESTful web APIs. I learned a lot about Django REST framework with the help of this book. This book had a chapter about Blog API where it showed how to build an API. It showed step by step process from initial setup to serializers and views. After reading that chapter and taking it as a reference to build my own API, I was successfully able to build my own API for the project. I learned about serializers which not only transforms data into JSON but can also specify which fields to include or exclude. I also learned about various generic views that Django REST Framework had.

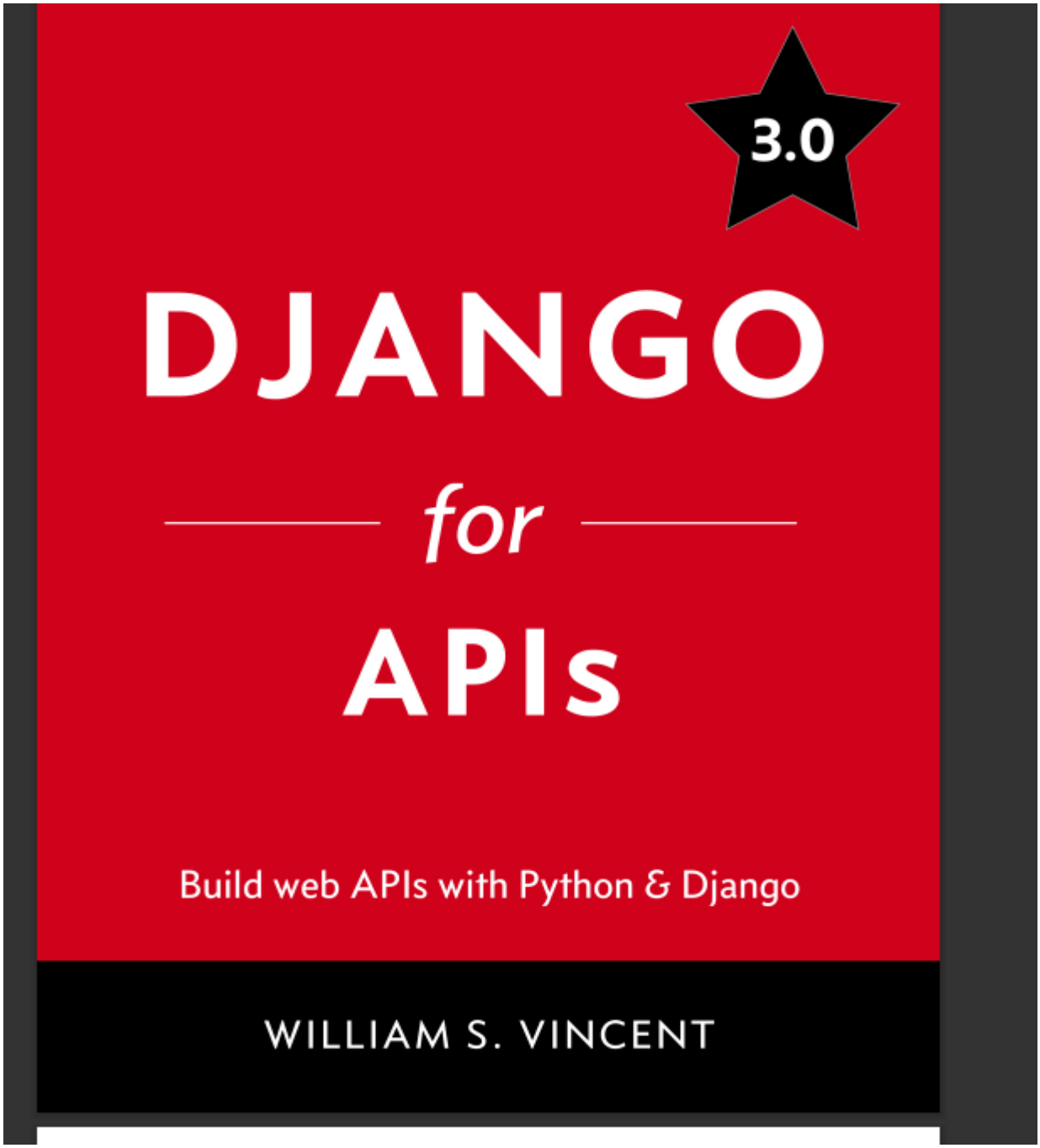

Figure 15 Screenshot of Django for APIs

And here is the picture of the book that helped me in learning how to make API for my project.

Evidence of Issue tracking

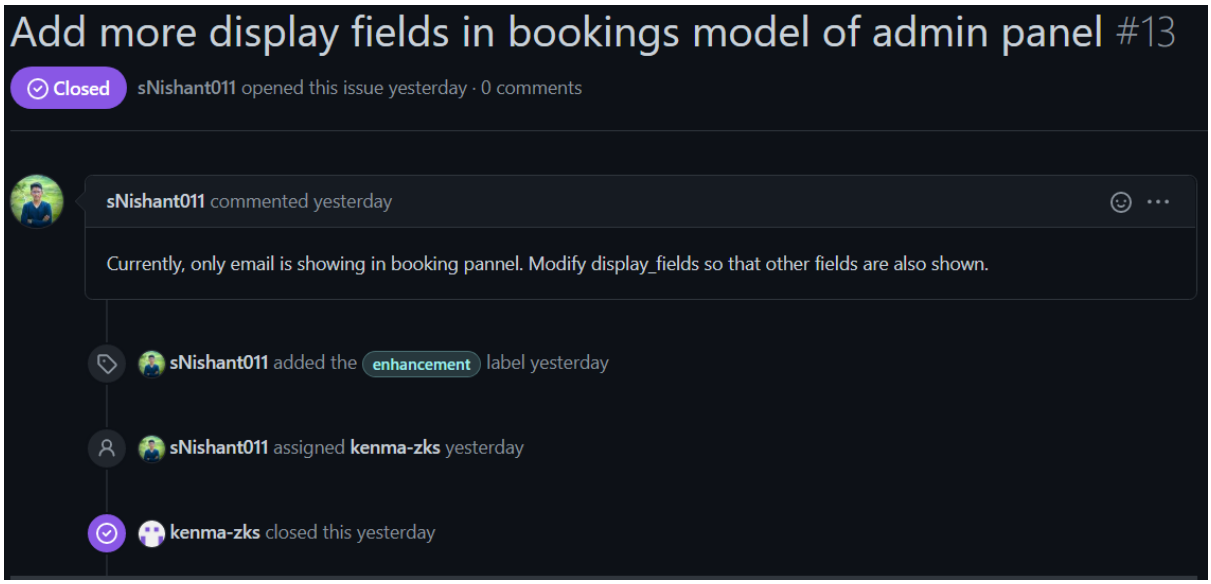

Figure 16 Issue about having less display fields

Figure 17 Issue regarding front-end developer not being able to login in admin panel

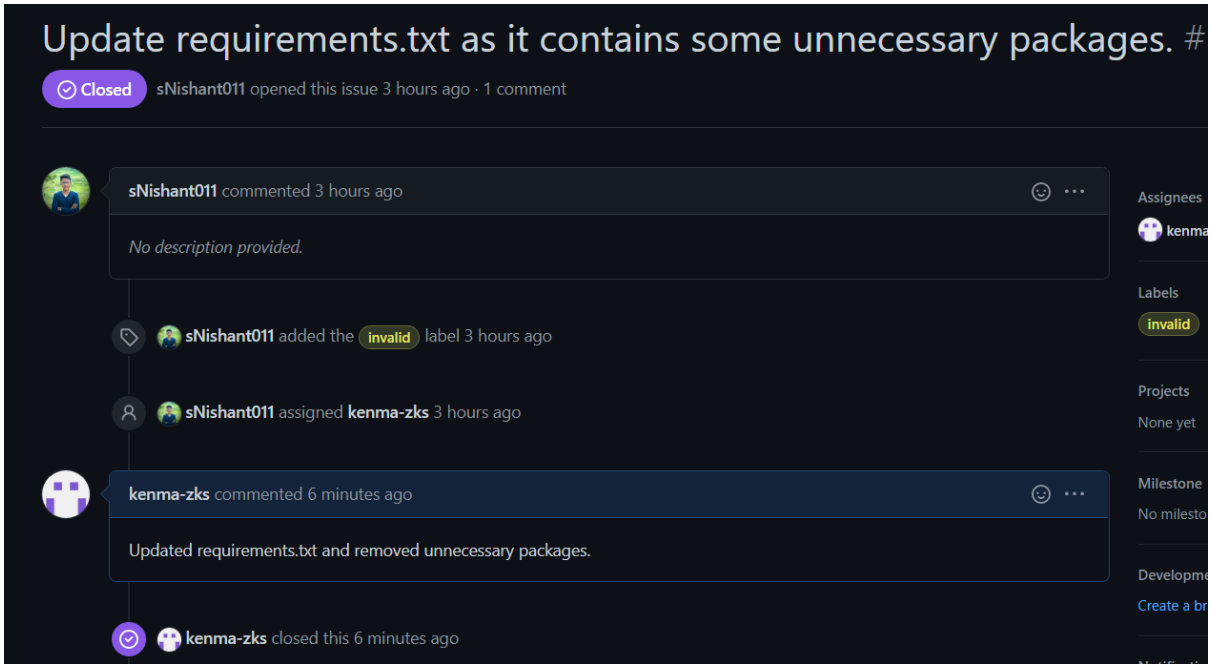

Figure 18 Issue regarding requirements.txt having unnecessary packages

The first issue I had was booking models of admin panel not having enough display fields. I only displayed email field. To fix this issue I created a new list called list_display inside BookingAdmin class. Inside the list I added four new fields which I had previously created in models.py such as first name, last name, phone number and booked packages. With this all these fields could be displayed in the admin panel of bookings. Then I closed this issue.

In the second issue, I had forgot to inform others about the username and password of the superuser I had created in Django. So, after receiving the issue I messaged him about the username and password of admin through comment and closed the issue.

In the third issue, the requirements.txt file had some unnecessary files or files which I was no longer using for the project. I had deleted the packages inside folder, but I had forgot to update the requirements.txt file. This lead to requirements.txt having unnecessary files that the project was no longer using. After receiving the issue, I updated the requirements.txt file with pip freeze > requirements.txt which updates the requirements.txt file. After updating the file, I had closed the request and informed about updating the file through comment.

References

Data flair. (n.d.).

Dhaduk, H. (2021, February 10). *simform*. Retrieved from https://www.simform.com/blog/django-vs-php/

Duggal, A. (2021, November 21). *HEVO*. Retrieved from https://hevodata.com/learn/sqlite-vs-mysql/