1. Creating an virtual enviroment (git bash)

* Command creating env: python -m venv enviroment\_name
* Activating virutal enviroment using command: source env/Scripts/activate
* To see all installed packages we use command: pip freeze
* To deactivate virtual env we use command: deactivate

1. Intsaling Django in venv (git bash; Django version 4.2.2)

* To install Django we use command: pip install django
* To create django project: django-admin startproject greatkart . (with the dot in the end)
* To activate django server: python manage.py runserver

1. Running http respone (visual studio code)

* Before changing anything in the files, we need to change python intepreter in VSC to the one created as virtual enviroment.
* Adding to file urls.py one more url link and importing file views
* Creating folder views.py in which is placed the template to the site from bootstrap.

1. Importing bootstrap templates

* Importing template files to main folder
* Creating folder templates in main project folder, in which is created a file named home.html
* Changes in the folder greatkart, file settings.py. Setting the variable TEMPLATES argument DIRS to ‘templates’ incializing the previously created folder
* Copying the code from index.html to home.html
* Creating folder named static and moving to it all folders from template (css, fonts, images, js)
* Creating new variables in file settings.py:
* STATIC\_ROOT which refers to the static folder
  + STATICFILES\_DIRS which refers to the static folder
* Running command „python manage.py collectstatic” in gitbush which will copy all files from the static folder ale create another one in the root folder.

1. Dividing sections from html file

* Loading the static folder to home.html file using DTL (Django Template Language). DTL is used for generating dynamic content and dividing parts of code into sections which are then conected with each other through refrences.
* Method I:
  + Adding a line of code {% load static %} in the beginning of the html file
  + Changing all the refrences to css links and js scripts. The pattern is {% static ‘link\_or\_script\_path’ %}.
  + Changing all the refrences to images. The pattern is {% static ‘image\_path’ %}.
* Creating another html file named base in folder templates.
* Extracting to base.html the head and header section of home.html.
* Joining two html files with {% extends ‘base.html’ %} placed in the home.html
* Using {% block content %} and {% endblock %} to place in right place the code from the other file.
* Method II:
* Creating a folder named inlcudes
* Placing two files footer.html and navbar.html
* Cutting out the footer and nav sections from html files and placing them acordingly as the names of files
* Placing {% include 'includes/navbar.html' %} and {% include 'includes/footer.html' %} before and after the section block in the base.html file
* \* Adding to some files {% load static %} that need refrence to this folder.

1. Creating app (django-admin-panel)

* Creating a folder named category with „python manage.py startapp category” (bash). Its a Django framework command that generates a structure for new app.
* Registering the category app in INSTALLED\_APPS list in file settings.py
* Django.db models enables to define datamodels that we can use to keep, download, modify or delete data in database. It enables interaciton with database and it lets to manipulate data in Django app.
* Creating a category class in models.py in order to keep data in database
* Importing and registering model in admin.py
* Installing pillow framework for images
* Command „python manage.py makemigrations” creates migration files. It is used to propagate changes in the models into the database schema.
* Running created migration file „python manage.py migrate”
* Creating a superuser to enable entering into /admin site with command „winpty python manage.py createsuperuser” (bash)
* To change the spelling of „Categorys” to „Categories” in admin panel, we create meta class in models.py

1. Custom user model

* Creating new user app named accounts „python manage.py startapp accounts” (bash)
* In the main folder greatkart, we add in file settings.py another argument to variable INSTALLED\_APPS
* Creating accounts models in file models.py
* Importing AbstractBaseUser to assure basic user model implementation, and BaseUserManager to manage those models
* Creating mandatory fileds such as join date or last login date
* Creating superadmin model in file models.py
* Creating if cases for mandatory fields
* Creating user creator function
* Creating superuser creator function
* Setting superuser privileges to True
* Adding a new variable AUTH\_USER\_MODEL to file settings.py in greatkart folder with the path to the Account class
* Registering the model in admin.py file in accounts folder
* Deleting the previous database because it may contain old data that would result in a conflict with data
* Making migrations to the app with new functions