


## Support chat

### Practicum

Hello! This is the Practicum Support Team. If you have questions or something to share, write to us directly in this chat window or select another option by clicking “Alternative contact methods” below. We are now back to supporting you 24/7 

Alternative contact methods

## Description

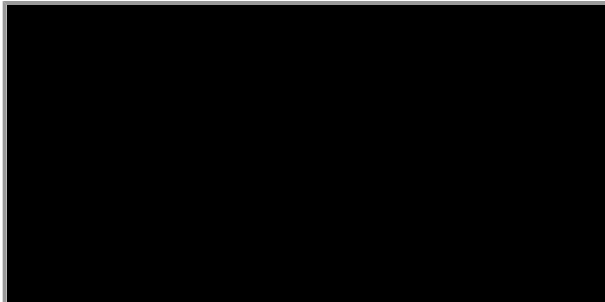
### Review

#### Project Description

Congratulations! You’ve completed the section on Software Development Tools. Now it’s time to apply the knowledge and skills you’ve acquired to a project: building and deploying a web application dashboard to a cloud service. Once you’ve finished the project remember to send your work to the project reviewer for assessment. They’ll give you feedback within 24 hours. Use the feedback to make changes, then send the new version back to the project reviewer. You might get further feedback on the new version. This is completely normal. It’s not uncommon to go through several cycles of feedback and revision. Your project will be considered complete once the project reviewer approves it.

#### Project description

The focus of this project is to provide you with additional practice with common software engineering tasks. These will augment and complement your data skills, and make you a more attractive job candidate to potential employers. The tasks are: creating and managing python virtual environments, developing a web application, and deploying it to a cloud service that will make it accessible to the public. In this project, we are providing you with a dataset you are already familiar with: a dataset of car sales advertisements. However, in this project, the focus will not be on the dataset or the analysis, and thus you are free, and even encouraged, to choose any dataset that you want. The project is split into several steps that replicate the process described in [one of our blog posts](#).



Demo of the web application you’ll build in this project

#### Instructions for completing the project

## Step 1. Project Prerequisites

1. Create an account on [github.com](https://github.com)

2. Create a new git repository with a `README.md` file and a `.gitignore` file (choose a Python template).
3. Install at least the following packages: `pandas`, `streamlit`, `plotly-express`. Feel free to install more packages if you want to implement additional features in your web app.
4. Create an account on [render.com](https://render.com), link it to your GitHub account
5. [Install VS Code](#), load the project directory and set the Python interpreter to the one used by the virtual environment

## Step 2. Download the data file

1. [Download the car advertisement dataset](#) (`vehicles_us.csv`) or find your own dataset in a CSV format
2. Place the dataset in the root directory of the project

Note: for project ideas and inspiration, check out [this video](#).

## Step 3. Exploratory Data Analysis

1. Create an `EDA.ipynb` Jupyter notebook in VS Code
2. Save the notebook in the `notebooks` directory of your project
3. Perform some basic exploratory analysis of the dataset in the notebook
4. Create a couple of histograms and scatterplots using `plotly-express` library

Note:

- if you are using the car advertisement dataset, it won't be sufficient to simply recreate the plots described in the blog post to complete the project. You'll have to get creative and come up with your own plots and histograms.
- it's often very convenient to experiment with data visualizations in Jupyter, and then copy-paste code into a web application file later

## Step 4. Develop the web application dashboard

1. Create an `app.py` file in the root of the project's directory
2. Import `streamlit`, `pandas` and `plotly-express`
3. Read the dataset's CSV file into a `DataFrame`
4. Create and/or copy from the Jupyter notebook:
  - at least one `st.header` with text
  - at least one `plotly-express` histogram using `st.write` or `st.plotly_chart`
  - at least one `plotly-express` scatter plot using `st.write` or `st.plotly_chart`
  - at least one checkbox using `st.checkbox` that changes the behavior of any of the above components
5. Don't forget to update the `README` file when you are done. It should contain some basic description of the project and instructions on how other people could launch your project on their local machine if they wanted to.
6. **IMPORTANT: Replit will fail to build your project unless all project files are present in your GitHub repository.** Therefore, you must commit and push file changes to your repository when you're done with your work.

Notes:

- as you develop your application by adding a new Streamlit component, you can run `streamlit run app.py` command from the terminal to see what the result looks like
- as you reach some milestones in the application development (e.g. you add a working component and the application runs without errors), it's a good practice to commit and push your work to a remote repository on GitHub. So don't forget to write a meaningful commit message!

## Step 5. Deploy the final version of the application to Render

1. To make streamlit compatible with render, add a streamlit configuration file to your git repository at `.streamlit/config.toml` with the following content:

Copy code TOML

```
[server]
headless = true
port = 10000

[browser]
serverAddress = "0.0.0.0"
serverPort = 10000
```

It'll tell Render to look in the right place to listen to your streamlit app when hosting it on its servers.

2. Open your account on [render.com](https://render.com) and create a new web service:



3. Create a new web service linked to your Github repository:



4. Configure the new Render web service. To your **Build Command**, add

Copy code BASH

```
pip install streamlit & pip install -r requirements.txt
```

To your **Start Command**, add: `streamlit run app.py`. It should look like this: Deploy to Render, wait for the build to succeed:



5. Deploy to Render, wait for the build to succeed:



6. Verify that your application is accessible at the following URL: `https://<APP_NAME>.onrender.com/`

**Note:** it can take several minutes after a succesful deployment for the app to be available online on a free tier. Also note that apps go “asleep” after being inactive for a few minutes. If so, just load and refresh your app a few times for it to get awoken.

## How to submit my project:

You'll need to submit a link to your GitHub repository. Please also add the URL of your app on Render to your project's `README.md`

## How will my project be evaluated?

We've put together some project assessment criteria. Read over them carefully before you make a submission. Here's what project reviewers look for when assessing your project:

- Does the project repository contain at least the following files?

## Copy code

# The minimal expected project structure

\$ tree

```
.
├── README.md
├── app.py
├── <name_of_your_dataset>.csv
├── notebooks
│   └── EDA.ipynb
├── .streamlit
└── config.toml
```

- Is the web app accessible via a browser?
- Does the web app contain the following?
  - at least one header with text
  - at least 1 histogram
  - at least 1 scatter plot

Submit   Next

Step 1/2

Send the link to your assignment

Submit

Submitting assignment

Review in progress

## common.callbackForm.dataScientist.title

common.callbackForm.dataScientist.description.afterhours

common.callbackForm.dataScientist.input.userName.label

common.callbackForm.dataScientist.input.phoneNumber.label

common.callbackForm.dataScientist.userAgreement

common.callbackForm.dataScientist.actionButton.label

common.callbackForm.dataScientist.delimiter

common.callbackForm.dataScientist.footer.description.primary

common.callbackForm.dataScientist.footer.description.secondary

common.callbackForm.dataScientist.socials.common.callbackForm.dataScientist.socials.list.title