# Online Pyton Interpreter

Language – Python

Framework – Django

Submitted To – Mr. Sagar Pande

Group Members:

1) Bhaskar Mahto – 11917448

2) Abunachar Yeahhia Mazumder - 11917210

Subject – Python Programming

Code - INT-213

**Summary**

There are many resource out there but lack of practice makes thing harder to keep in mind for long term or people don’t know where to get started and always lose track form what they are aiming for.

Analyzing this we have created this Project which is a Online Python Compiler website Using Django Framework and Python which provide everyone a open source knowledge resource to learn new Topic of python and practice the topic what they have learn in the same platform.

This a Simple Minimalist site provides a Friendly user interface for the user to interact. There are may many modules Which user can access, like Learn Module in which user can get beginner and advance type question.User can select the Beginner or Advance as per his/her level and see different types questions.To access the answers of the Question User have to login or sign up if they don’t have an account. After Successful login the user will be Automatically redirected to the answer page.

The login/sign up system is very save with Django password hashing and salting in password, which cannot be decoded without exact key,Salting code and hashing algorithms used.The data are stored in SQLite3 Database.

If user Want to run the program he/she can copy the code directly go to the Compiler module to run the code. The Compiler module has a simple user interface with all placeholder for new user to navigate. There are 3 main panel which are code panel,input panel,terminal/output panel. The code panel in which user can type the code which they want to run ,next is the input panel it the program contain any input syntax they can use the input panel to give all the required input else it can be left empty and the last output panel which is an read only panel which prints all the output and the total time of execution and total memory consumed by the program. These module are achieved using Hacker Rank API module which we have implemented in out project.

The main contributor of this project are **Abunachar Yeahhia** ( knight-byte on GitHub ) and **Bhaskar Mahto** ( arbkm22 on GitHub)

**Index**

* Introduction
* Review of Literature
* Project Contribution
* Material ans Methods
* Result
* Conclusion
* Bibliography

**Introduction**

This Project which is a Online Python Compiler website Using Django Framework and Python which provide everyone a open source knowledge resource to learn new Topic of python and practice the topic what they have learn in the same platform.

This a Simple Minimalist site provides a Friendly user interface for the user to interact. There are may many modules Which user can access, like Learn Module in which user can get beginner and advance type question.User can select the Beginner or Advance as per his/her level and see different types questions.To access the answers of the Question User have to login or sign up if they don’t have an account. After Successful login the user will be Automatically redirected to the answer page.

The login/sign up system is very save with Django password hashing and salting in password, which cannot be decoded without exact key,Salting code and hashing algorithms used.The data are stored in SQLite3 Database.

If user Want to run the program he/she can copy the code directly go to the Compiler module to run the code. The Compiler module has a simple user interface with all placeholder for new user to navigate. There are 3 main panel which are code panel,input panel,terminal/output panel. The code panel in which user can type the code which they want to run ,next is the input panel it the program contain any input syntax they can use the input panel to give all the required input else it can be left empty and the last output panel which is an read only panel which prints all the output and the total time of execution and total memory consumed by the program. These module are achieved using Hacker Rank API module which we have implemented in out project.

**Review of Literature**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language" \o "Interpreted language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language" \o "High-level programming language) and [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language" \o "General-purpose programming language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum" \o "Guido van Rossum) and first released in 1991, Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability" \o "Code readability) with its notable use of [significant whitespace](https://en.wikipedia.org/wiki/Off-side_rule" \o "Off-side rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct" \o "Language construct) and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming" \o "Object-oriented programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers" \o "Programmers) write clear, logical code for small and large-scale projects.

Python is [dynamically typed](https://en.wikipedia.org/wiki/Dynamic_programming_language" \o "Dynamic programming language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)" \o "Garbage collection (computer science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms" \o "Programming paradigms), including [structured](https://en.wikipedia.org/wiki/Structured_programming" \o "Structured programming) (particularly, [procedural](https://en.wikipedia.org/wiki/Procedural_programming" \o "Procedural programming)), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming" \o "Object-oriented programming), and [functional programming](https://en.wikipedia.org/wiki/Functional_programming" \o "Functional programming). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library" \o "Standard library).

Python was created in the late 1980s as a successor to the [ABC language](https://en.wikipedia.org/wiki/ABC_(programming_language)" \o "ABC (programming language)). Python 2.0, released in 2000, introduced features like [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension" \o "List comprehension) and a garbage collection system with [reference counting](https://en.wikipedia.org/wiki/Reference_counting" \o "Reference counting).

Python 3.0, released in 2008, was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility" \o "Backward compatibility), and much Python 2 code does not run unmodified on Python 3.

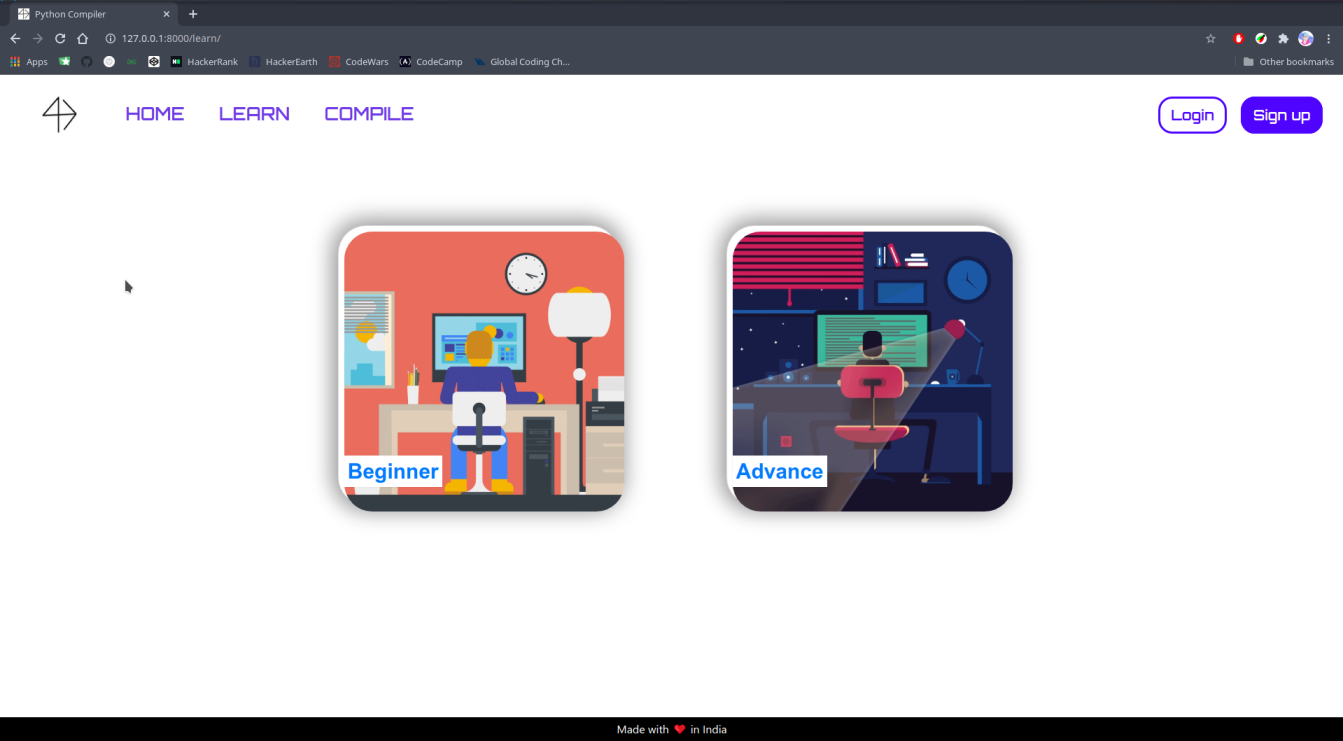
Django is a [Python](https://en.wikipedia.org/wiki/Python_(programming_language)" \o "Python (programming language))-based [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software" \o "Free and open-source software) [web framework](https://en.wikipedia.org/wiki/Web_framework" \o "Web framework) that follows the model-template-views (MTV) [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern_(computer_science)" \o "Architectural pattern (computer science)).It is maintained by the [Django Software Foundation](https://en.wikipedia.org/wiki/Django_Software_Foundation" \o "Django Software Foundation) (DSF), an American independent organization established as a [501(c)(3)](https://en.wikipedia.org/wiki/501(c)(3)" \o "501(c)(3)) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability" \o "Reusability) and "pluggability" of components, less code, low coupling, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don't_repeat_yourself" \o "Don't repeat yourself). Python is used throughout, even for settings files and data models. Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete" \o "Create, read, update and delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Type_introspection" \o "Type introspection) and configured via admin models.

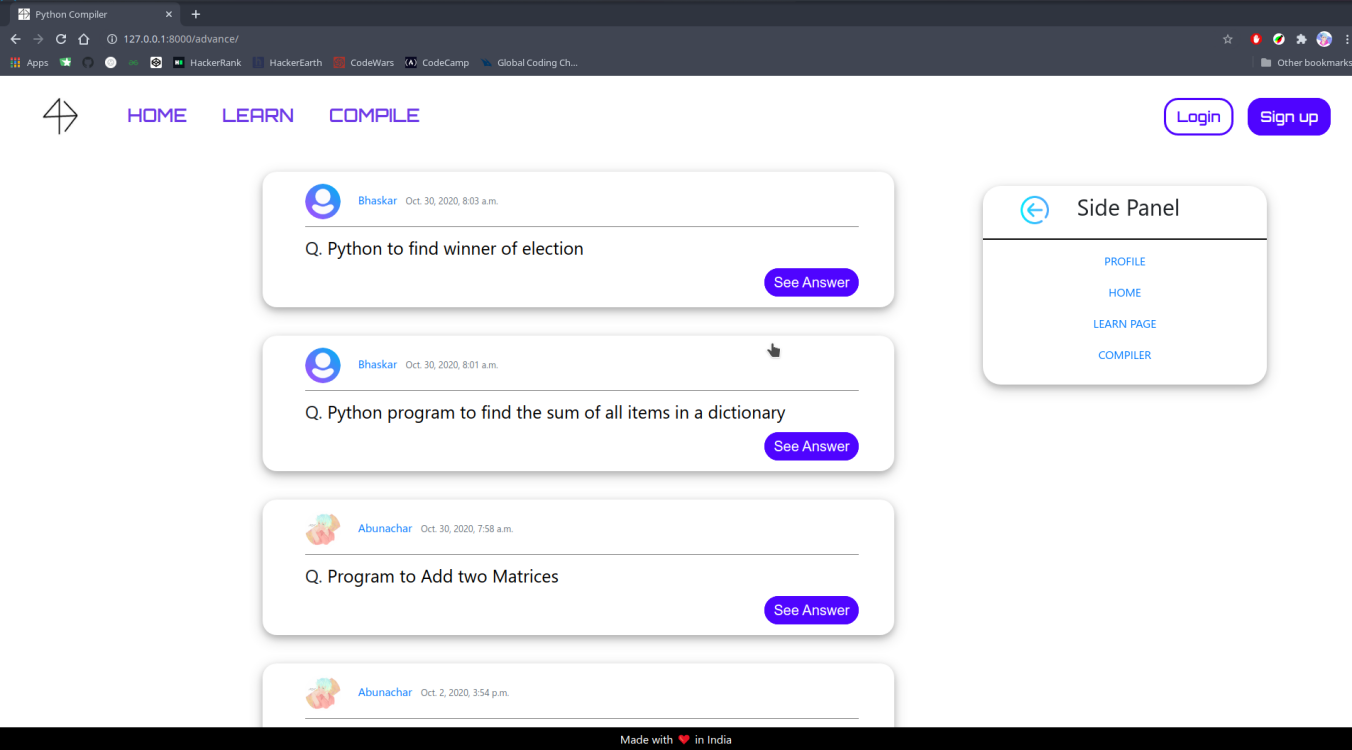
**HOME PAGE**



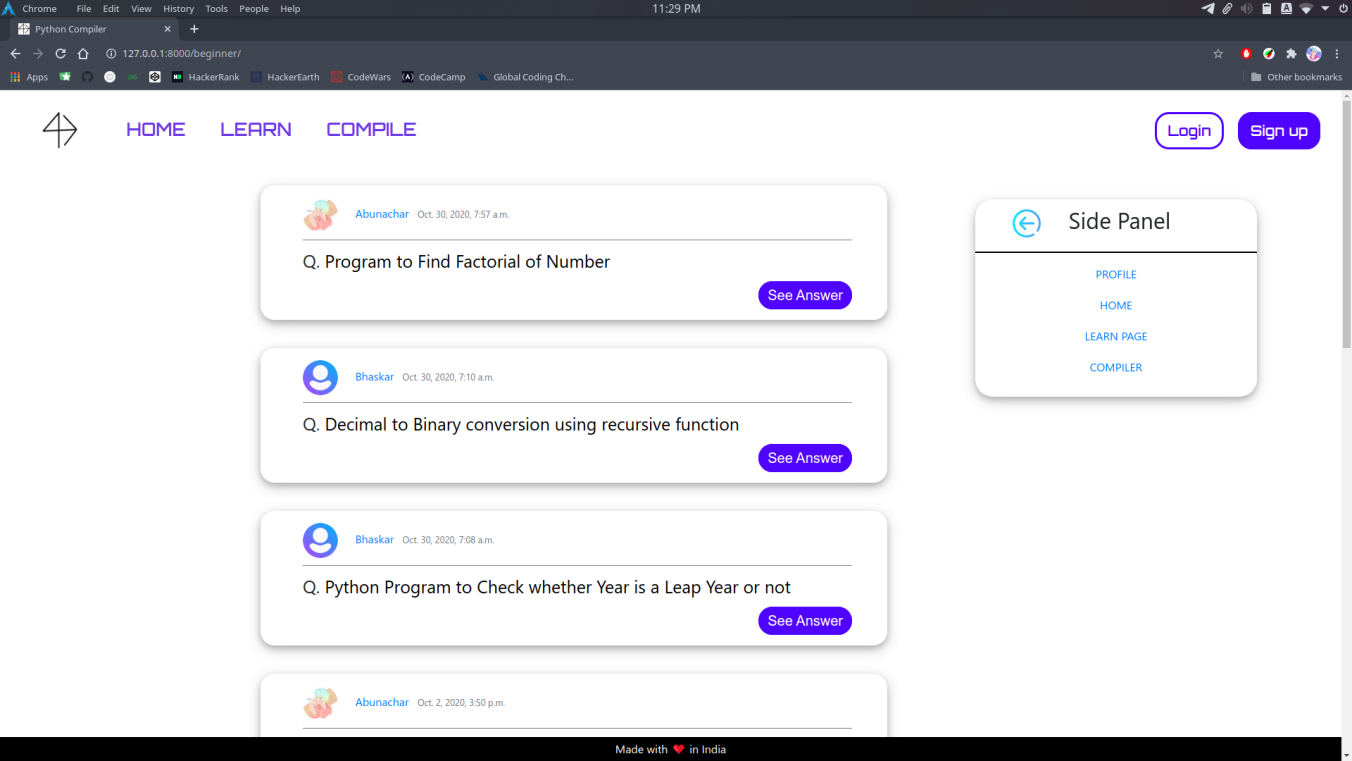
**LEARN PAGE**

****

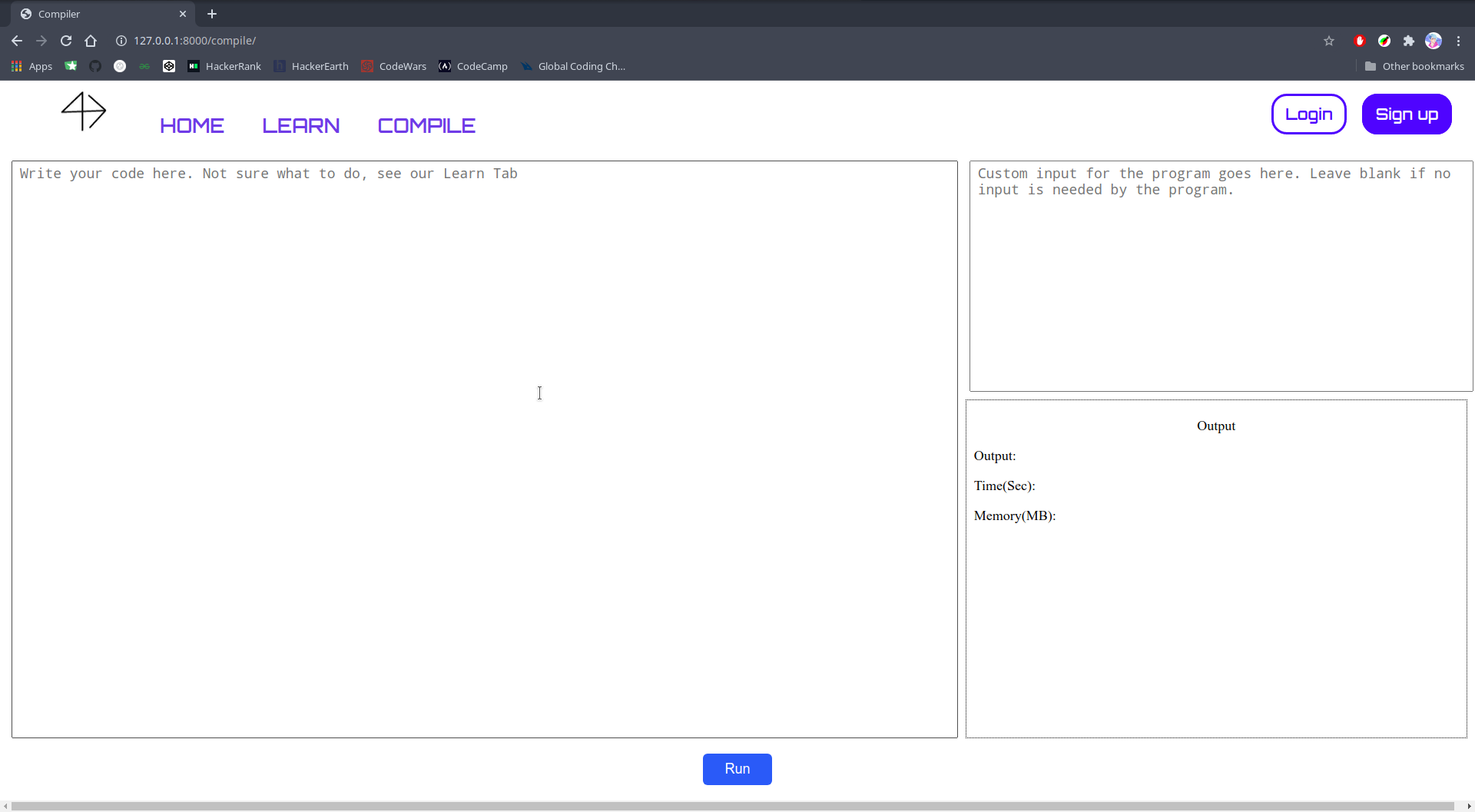
**BEGINNER POST PAGE**

****

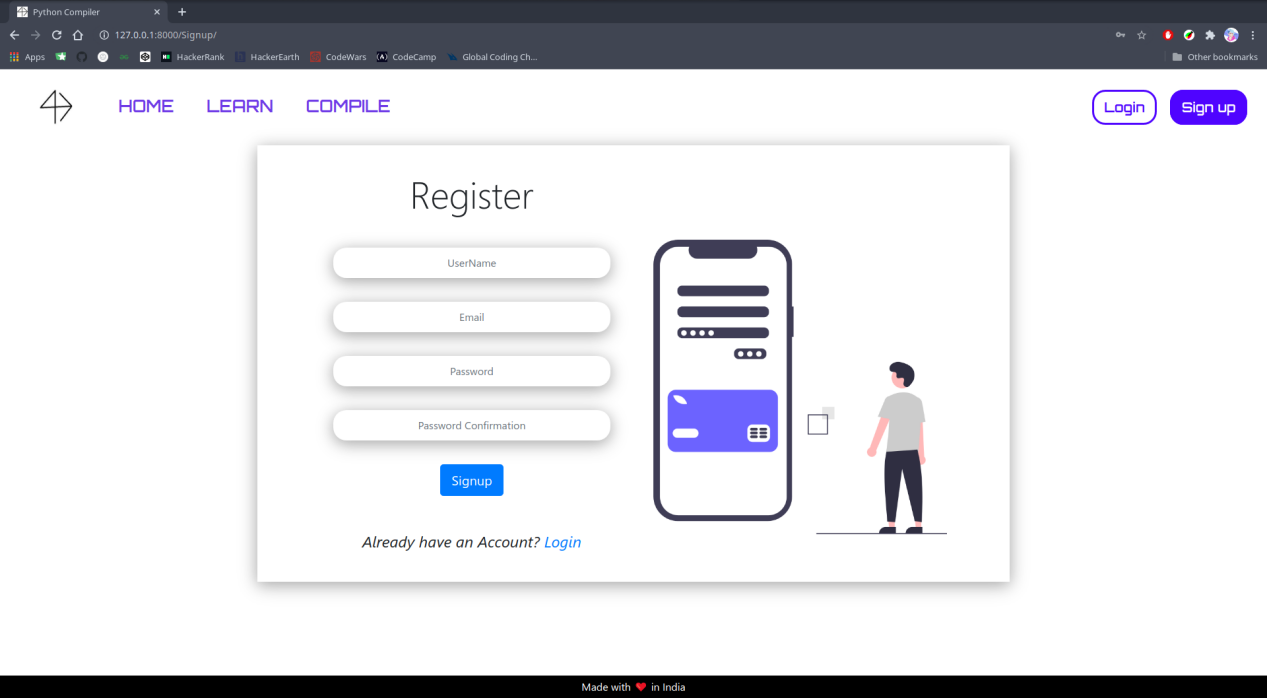
**ADVANCE POST PAGE**

****

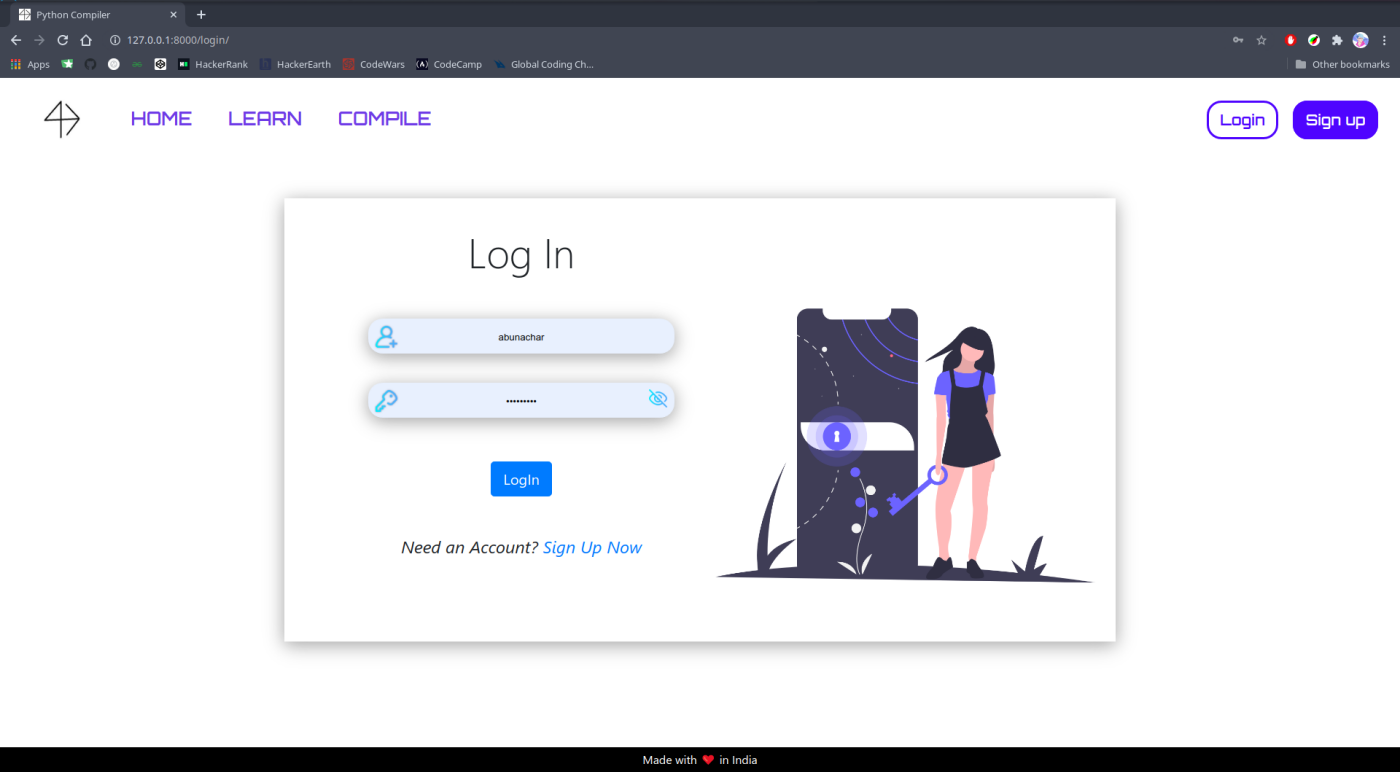
**COMPILER PAGE**

****

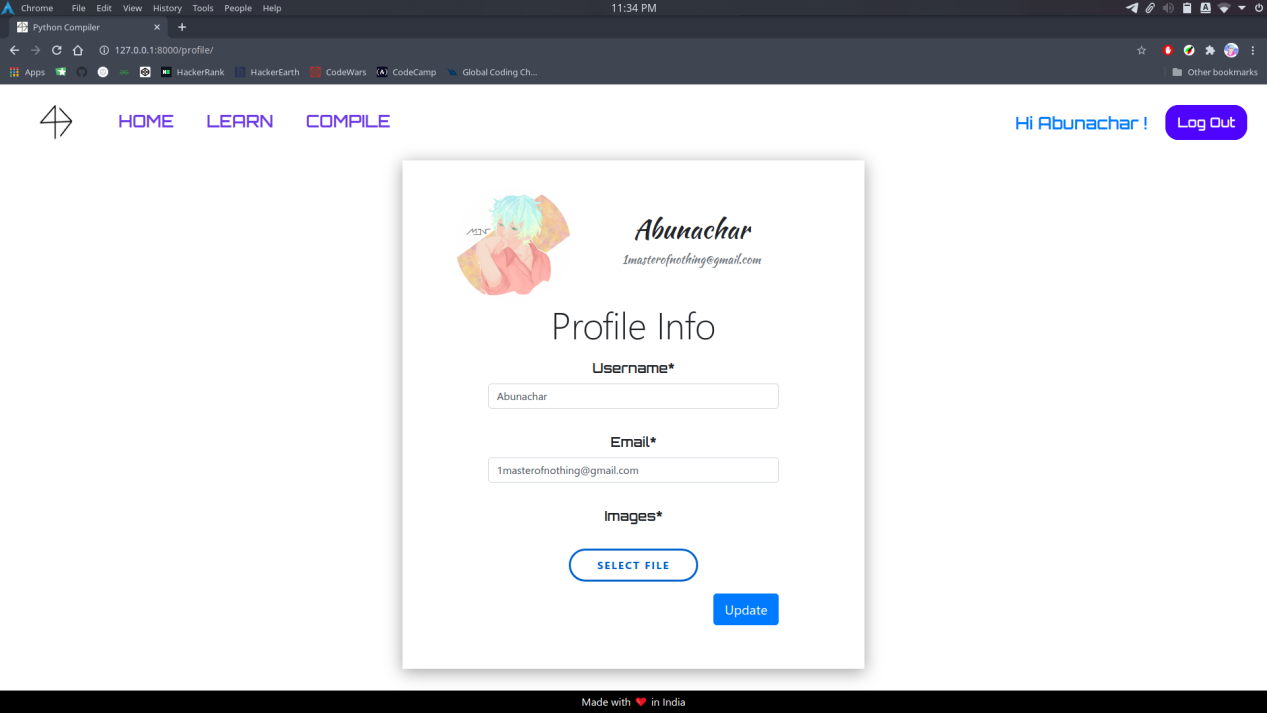
**SIGIN UP PAGE**

****

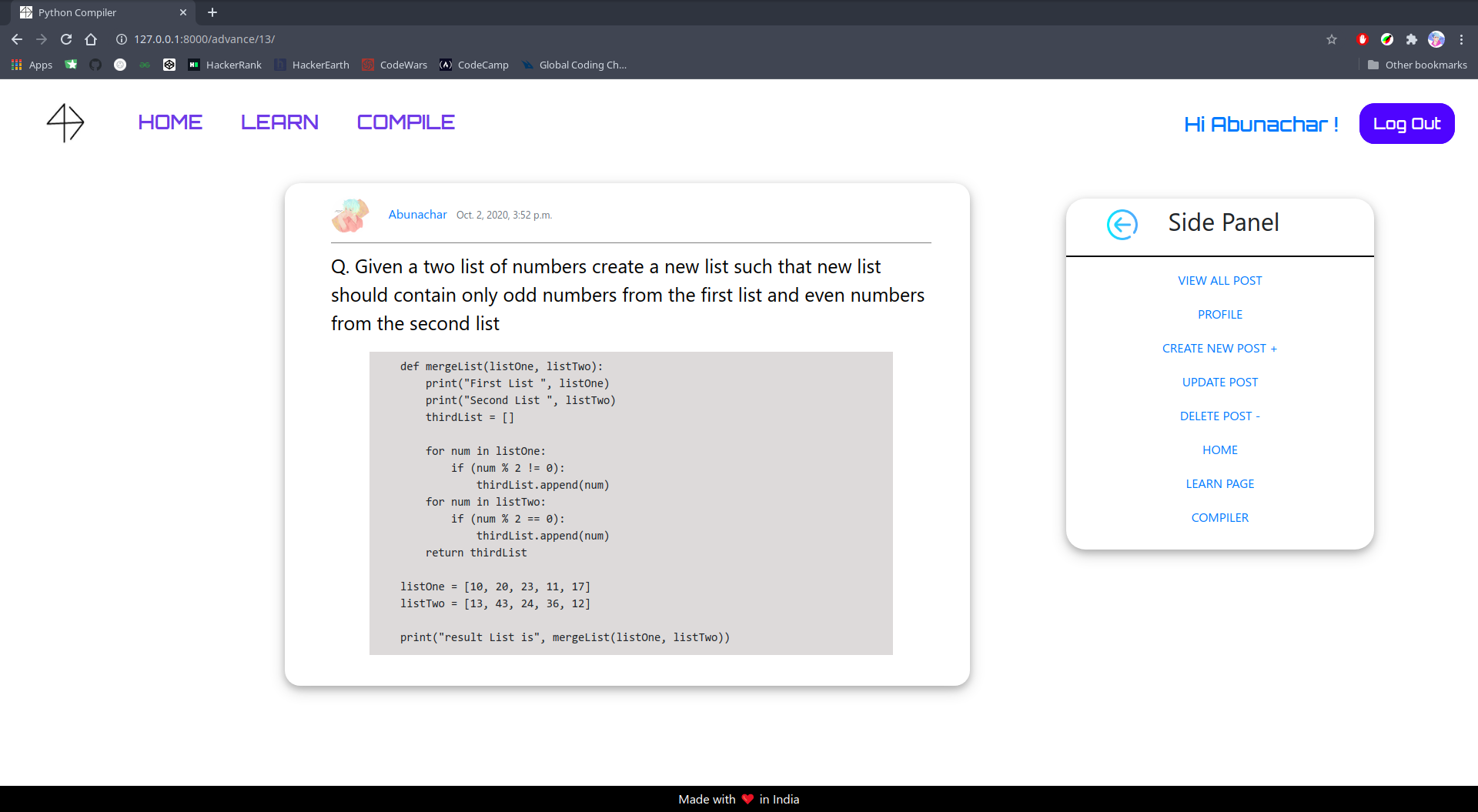
**LOGIN PAGE**

****

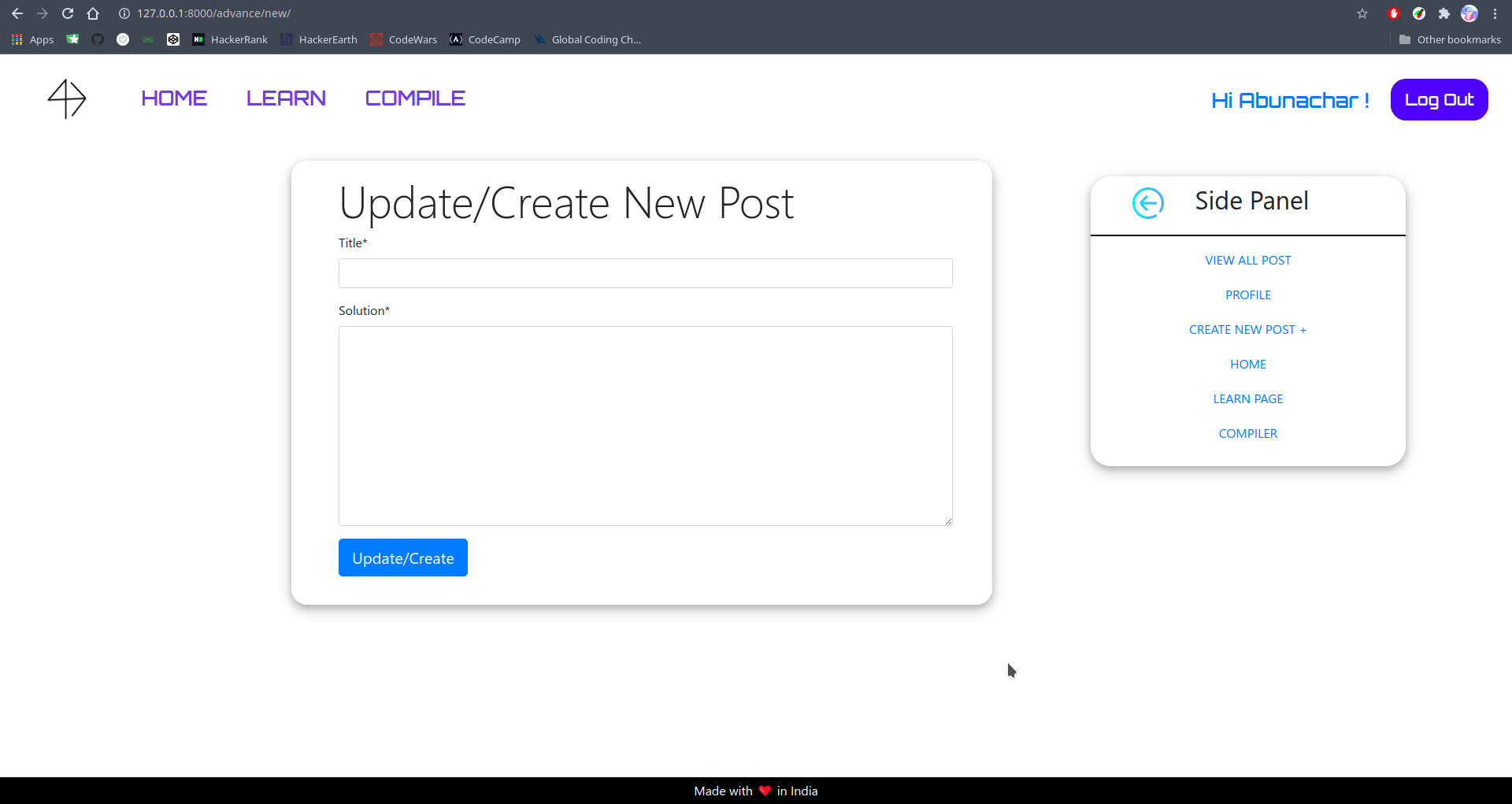
**USER PROFILE PAGE**

****

**DETAILED POST/ANSWER PAGE**

****

**CREATE / UPDATE PAGE**

****

MATERIALS & METHODS

The following list showcases all the materials used for making the project i.e., Online Python Compiler.

* Django (v2.1): The main Python Framework, used to handle the Front-End as well as the Back-End of the project.
* requests(v2.24.0): Python library for handling all the HTTP requests made from the user side.
* djangorestframework(v3.12.1): Python module for django, to handle all the rest-api calls and POST/GET requests.
* django-crispy-forms(v1.9.2): Django module for better form customisation.
* HackerEarth API(v3): API for compiling and running code on HackerEarth servers instead of using user’s system.

**RESULT**

The final outcome of our project is a fully functional python interpreter that is not system dependent. It runs on the cloud and doesn’t require your system to have python installed. It can also work on handheld devices like Smartphones or even PSP’s (as long as the device have a compatible browser that supports ES6). The only drawback is the lack of dynamic programming, i.e., the user can’t interact with the system once the program has been submitted. If there are any inputs that is required for the program to run, then the user has to submit it beforehand.

Also, we have a integrated learning section, that provides with some good examples based on the level of experience of the user, i.e., beginner to advancewd. We have provided some common examples and their solution too.

We have also integrated a user database for more functionalities like the option to add your own set of examples so that other users can also benefit from it. To add or view answers, you need to be registered on the site.

To protect the server from heinous inputs, we have added CSRF\_Tokens, so that the security of the site remains uncompromised, along with the user data.

**DISCUSSION (including conclusion, recommendation)**

The final outcome of the program is a clean and proper working Python Interpreter. We have hosted the web app on Heroku. Since we believe that Open Source is the way of the future, the source code of the app is made publicly available on GitHub along with the API Token, so that our users, if they find any errors, or want to contribute to it, can do so freely without having to worry about the API Tokens (as it is very cumbersome to get the Tokens). If you want to make any improvements, or have any suggestions, fork the repo, and create a pull request. We will review and merge the request ASAP.

To run and test the app locally, follow the instructions on our GitHub readme.

**Recommendation**

Some users may want to use Flask instead of Django, as Flask is more lightweight and gives the user more power and authority over the app. To be honest, we intended to use Flask for this, but due to some difficulties, we had to settle in with Django. The main api handler can be found in the “views.py” inside “compiler” directory of the main program.

**Bibliography**

Web Links :

* <https://docs.python.org/3/library/index.html>
* <https://docs.djangoproject.com/en/3.1>
* <https://www.hackerrank.com/api/docs>

Book and Reference :

- Django for Beginners: Build website with python and Django by William S. Vincent