

---

# PYTHONIFY

## Practice Problem Tracker

pythonify

[Home](#) [Challenges](#) [Manage](#)

[Admin](#)

### Practice Set 2

< + Select Set ▾

Basic

#### P1: Hello World

The first program you will complete is a simple program that prints a phrase ("Hello World!") to the...

Basic

#### P2: Basic I/O

In the previous problem, you used the print command to print out text. You can perform 'user input' ...

Basic

#### P4: Conversion Problem

Write a conversion program to convert some number of one unit into another. The unit of choice can b...

Basic

#### P6: Grade Calculation

Write a grade calculator program that will compute and output the final grade of a student based on ...

Intermediate

#### P8: Simple Graphics Drawing

Please refer to PDF document for more information.

Create new practice problem

Developed by Shrish Mohapatra

Logged in as [smohapatra](#)

Pythonify is a secure web-based practice problem tracker application developed with Django. Key features include an authentication system with varying levels of access, ability to create/edit practice problems & sets, and various other features.

This report will discuss the purpose of the project, survey the various features included, and useful notes for future development & deployment.

**COMP1405 | SHRISH MOHAPATRA**

---

---

## TABLE OF CONTENTS

<b>PURPOSE</b>	<b>2</b>
Why Django	2
<b>HOW TO USE</b>	<b>3</b>
Authentication	3
Practice Problems	5
Miscellaneous	7
<b>DEVELOP</b>	<b>10</b>
Project Directory	10
Essential Files	11
Deploy	11
<b>CONCLUSION</b>	<b>12</b>
Scope for Future Development	12

---

## PURPOSE

The main objective of the project is to provide a platform where students may view and attempt various professor-created practice problems. This project can be broken down into three major components.

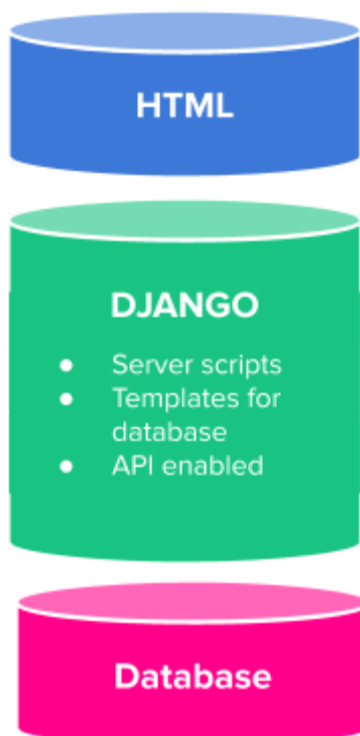
**Authentication.** This project requires a login system so that only designated users can access the site. Moreover, it requires a Sign-Up system where students can register an account, create their custom user tag, and use an admin-provided access code.

**Admin backend.** The project requires a “dashboard” where admins can view the progress of all the registered students. This is also where admins can approve pending user accounts, granting students access to the site.

**Practice problems.** The application must provide admins with the means of creating new practice problems and sets. For students, they should be able to view problems and mark them as complete to track their progress.

## Why Django

Django is an open source, Python-based web framework. The framework boasts high standards of security, speed, and scalability. Sites such as Instagram, Pinterest, and more, all use Django. The following is a schematic representation of a Django web application.



The web application starts with the database. While Django uses SQLite by default for development, one could use a range of database options- MySQL, PostgreSQL and most of the contemporary databases (<https://docs.djangoproject.com/en/2.2/>). The Django framework itself contains web server scripts which communicate with the database to grab information. These scripts and their purposes will be described in greater detail in the **Develop** section. The application then presents this information through HTML and CSS.

It is worth noting that Django is not the only web framework for Python. While there are countless modules which support the creation of web applications, such as Flask, Django comes equipped with tools for large scale applications. User authentication, administrative backend, SMTP server support, are just a few of the many features which Django comes with.

---

## HOW TO USE

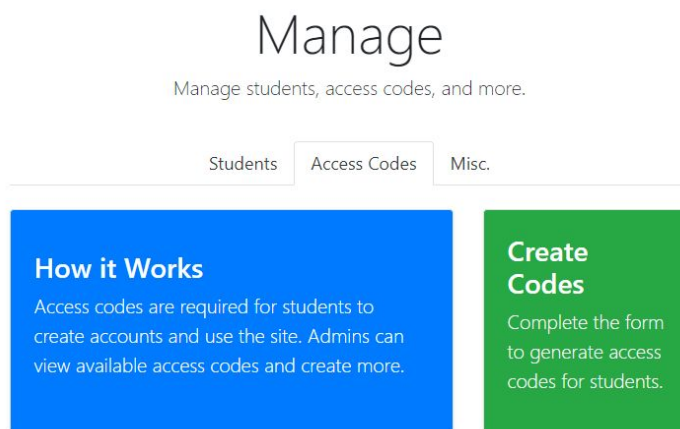
Pythonify comes with a plethora of features. The following will describe the authentication process, practice problem viewer, and miscellaneous capabilities (in that order). Each feature will be described from both the student and admin perspective, to provide a more cohesive understanding.

### Authentication

The general process for authentication is that the admin generates & provides access codes, students sign up, admins approve pending accounts, students access the site.

#### ADMIN SIDE

1. Login using admin credentials.
2. Select **Manage** from the dashboard.
3. Select the *Access Codes* tab on the **Manage** page.
  - a. Here you can review the purpose of access codes
  - b. In short, students will use the codes to register for accounts
4. Select **Create Codes** and complete the form.
  - a. You can specify how many access codes you would like to generate
  - b. You can specify how many characters each code is (suggested: 12)
5. Distribute codes to students.



#### FUTURE DEVELOPMENT

A future feature could include an automatic distribution of access codes via emails. As mentioned before, Django provides SMTP server support. Unfortunately due to time constraints this feature was not implemented.

## STUDENT SIDE

6. Navigate to site and select **Sign Up**.
  - a. Input a username which they will use to login with
  - b. Input a usertag which other students will see them as
  - c. Input a password (note, there is no Forget Password feature)
  - d. Input assigned access code.
7. Student will be informed that their account is pending approval.
  - a. At this stage, students will not be able to access the site

## ADMIN SIDE

8. Login using admin credentials.
9. Select **Manage** from the dashboard.
10. Select the *Students* tab on the **Manage** page.
  - a. Admins can view a table of all students in the system
11. Admins can approve pending accounts
  - a. Students will be able to access the site once approved
12. Admins can reject pending accounts
  - a. These accounts will be deleted
  - b. Their access codes will be available again

# Manage

Manage students, access codes, and more.



Students

Access Codes

Misc.

View Students

Pending 0 Registered 1

Username	Usertag	Completed	Status
shrishm	stevejobs	0 / 7	Registered  

---

## Practice Problems

Viewing and managing practice problems is the primary feature for Pythonify. Admins are able to create practice problems and sets, while students are able to view and complete problems.

### CREATING PRACTICE SET (ADMIN)

1. Login using admin credentials.
2. Select **Challenges** from the dashboard.
3. Click the **+** button in the top right corner.
4. Complete the form
  - a. Select the corresponding course
  - b. Enter the name of the set (ex. Practice Set 2)



### CREATING PRACTICE QUESTION (ADMIN)

5. Navigate to the **Challenges** page.
6. Select *Create new practice problem*.
7. Complete the form
  - a. Specify question #
  - b. Specify prompt name (ex. Search Engine)
  - c. Specify difficulty (Basic, Intermediate, Advanced)
  - d. Specify practice set.
  - e. Specify description (leave it blank to inform students to refer to PDF)

Once you have created a practice set and practice questions, you will be able to view them on the **Challenges** page. You can filter through different practice sets by clicking the **Select Set** button in the top right corner. You can then select any of the practice problem cards to view them.

**Practice Set 2**

**Basic**  
**P1: Hello World**  
The first program you will complete is a simple program that prints a phrase ("Hello World!") to the...

**Basic**  
**P2: Basic I/O**  
In the previous problem, you used the print command to print out text. You can perform 'user input' ...

**Basic**  
**P4: Conversion Problem...**  
Write a conversion program to convert some number of one unit into another. The unit of choice can b...

**Basic**  
**P6: Grade Calculation**  
Write a grade calculator program that will compute and output the final grade of a student based on ...

**Intermediate**  
**P8: Simple Graphics Drawing**  
Please refer to PDF document for more information.

Create new practice problem

<

+

Select Set ▼

Practice Set 1

Practice Set 2

---

## COMPLETING PRACTICE PROBLEMS (STUDENT)

8. Login using student credentials.
9. Select **Challenges** from the dashboard.
10. Select a practice problem, filter through sets if needed.
  - a. Students will view the problem's title, description, and difficulty
  - b. They can also filter through other problems in the same set

### P1: Hello World

<

Select Challenge ▾

Level: Basic

The first program you will complete is a simple program that prints a phrase ("Hello World!") to the console, which is the window in which the user interacts with the program. This program is simple enough that we can largely skip steps 1-4 of the above process. The algorithm for this program is one step: print "Hello World!" to the console.

11. Students will see the following for the first time
  - a. Students can provide feedback (satisfaction score)
  - b. Once they submit the form, the problem is marked as completed

Status*	Satisfaction*	
In Progress ▾	0/10 ▾	Submit

12. Students will see the average satisfaction for the question after.
  - a. Note: Admins will view this same page when they select problems

1 student has completed this challenge.

Average Satisfaction (90.00%)

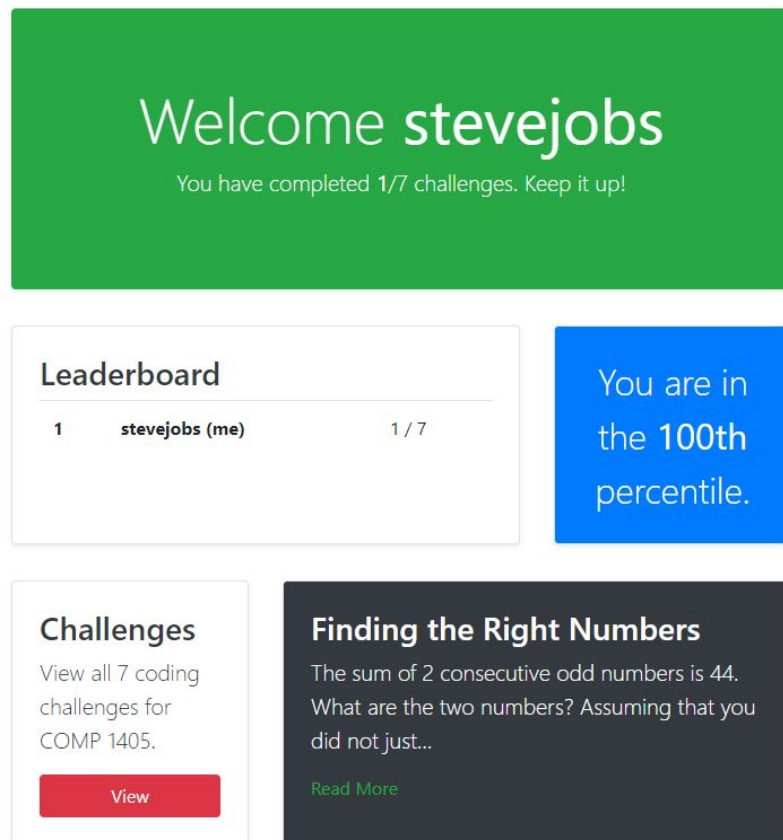


---

## Miscellaneous

The following features will include the dashboard features (for both Students and Admins), as well as the process of creating admin accounts.

### DASHBOARD (STUDENT)



The image to the left is the dashboard for students. This is the first page which students see after logging in.

The first **card** displays the student's usertag, as well as their practice problem completion record.

The second card displays a leaderboard, listing the top 3 students who have completed the most practice problems.

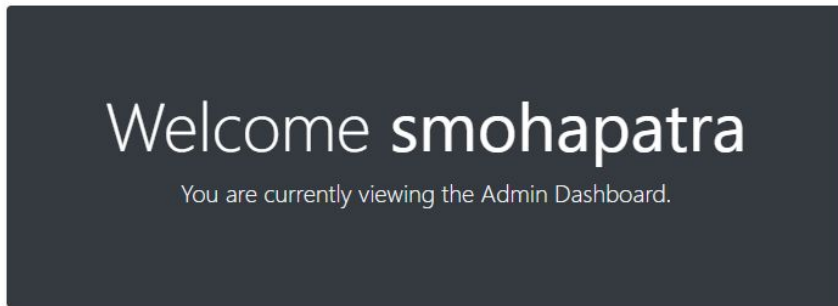
The third **card** displays the student's percentile.

The fourth **card** indicates the # of challenges available, allows students to view them as well.

The fifth **card** displays a "featured problem", and includes a brief description of the problem

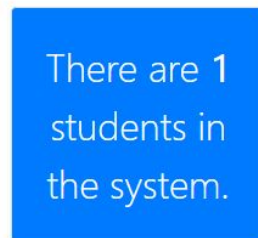
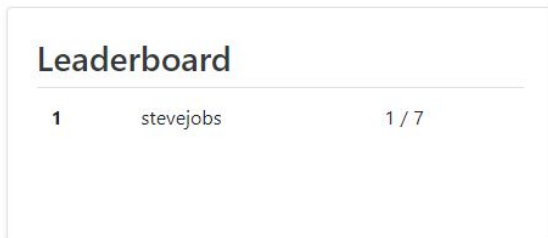


## DASHBOARD (ADMIN)

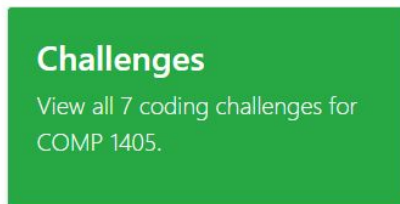
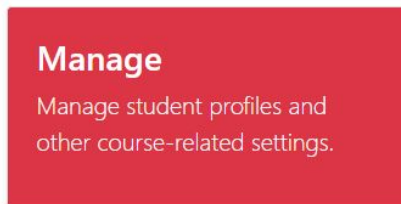


The image to the left is the dashboard for admins.

The first **card** displays the admin's username.



The second **card** displays a leaderboard, listing the top 3 students who have completed the most practice problems.



The third **card** displays the number of student accounts.

The fourth **card** provides admins with a portal to the Manage tab (viewing students, creating access codes).

The fifth **card** displays the number of coding challenges, allows admins to view and edit existing practice problems.

## CREATING ADMIN ACCOUNTS (ADMIN)

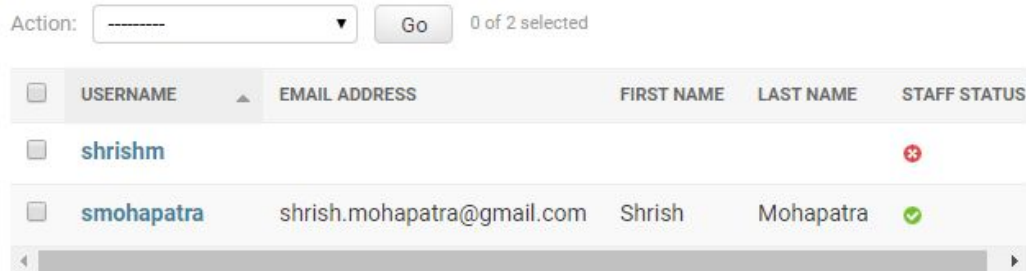
1. Login using admin credentials.
2. Navigate to the top right corner of the navbar, click **Admin**.

Admin

- a. This is Django's built-in backend site



3. Under *Authentication and Authorization*, select **Users**.
  - a. Admins will view a table of all users in the site (both Admin & Student)
  - b. Admins will be denoted with a ✓ in the *Staff Status* column



4. Click the **Add User** + button in the top right corner.
  - a. Provide a username, password
5. Upon clicking **Save**, admins will be redirected to another page.
  - a. The *Personal Info* section is not necessary to complete
  - b. Under the *Permissions* section, check all the boxes
    - i. Active, Staff status, Superuser status
6. Click **Save** to create the new admin account.

ADD USER +

---

## DEVELOP

Eventually it may be necessary for one to make development changes to the application itself. The benefit of using the Django web framework is that this is possible and relatively easy to do. Please note that this is **not** a detailed Django tutorial. For those interested in learning more, please refer to the various tutorials found online.

### Project Directory

```
/pythonify
|- core/
    |- migrations/
    |- templates/
        |- base.html
        |- index.html
        |- ...
    |- admin.py
    |- forms.py
    |- models.py
    |- urls.py
    |- views.py
    |- ...
|- pythonify/
    |- settings.py
    |- ...
|- db.sqlite3
|- manage.py
```

Understanding the project directory is essential for development with Django. Note that not all the files & folders are listed, just the most important.

As seen in the files, Django uses [Python](#), [HTML](#), and a database engine (default is [SQLite](#)).

The outermost folder (/pythonify) contains the database and the manage.py script (will explain later). It also contains the core/ folder, which contains the main app for the site, and the pythonify/ folder, which contains scripts important for the site itself (ex. settings.py).

In the core/ folder, one can find various files related to the app. These files will be explained in more detail.

There are also two folders; migrations/ folder and templates/ folder. The migrations/ folder contains a record of database changes made throughout development. If there are issues with the database, feel free to delete this folder's contents. The templates/ folder contains all the [HTML](#) files for the app. As mentioned before, a web application will present all information through HTML and CSS. Currently the app uses [Bootstrap 4](#) for CSS and Javascript.

---

## Essential Files

File	Purpose	File	Purpose
<code>admin.py</code>	Register database classes for backend.	<code>settings.py</code>	Site & project specific settings
<code>models.py</code>	Create database classes and fields (ex. Profile, Prompt)	<code>manage.py</code>	Will use terminal to call certain functions from this file (ex. <i>python3 manage.py runserver</i> )
<code>forms.py</code>	Which fields to show for forms (ex. Login, Signup, Feedback)	<code>db.sqlite3</code>	Database which stores practice problems, user login info, etc.
<code>views.py</code>	Core logic of the application. Functions to get information from the database.	<code>base.html</code>	This contains the navigation bar and initiates CSS & JS files.
<code>urls.py</code>	Used to map functions from <i>views.py</i> to URL links for HTML	<code>index.html</code>	The main page of the site. Includes the student dashboard.

## Deploy

There are various methods of deploying Django web applications. One of these methods includes deploying the application to a web server. This is done with the use of Apache to manage server configurations, virtualenv (or any other virtual environment package) to contain all python-related modules (ex. Django), and various other packages (mod\_wsgi). For a complete guide on how to deploy Django applications using Apache, please refer to this [article](#).

A factor to consider during deployment is the database engine. As mentioned previously, Django projects use the SQLite engine by default. SQLite is easy to work with however it does come with drawbacks. The engine does not support simultaneous changes to the database. This means that if two users were to login at the same time, the application would only be able to register one user before registering the other. This is not an issue for small-scale apps with few users, however with Pythonify this could be an issue. Fortunately, Django is compatible with various other engines such as MySQL and PostgreSQL, which support concurrency. Refer to this [article](#) if you are considering using PostgreSQL and are planning to deploy the app on a Linux server.

A final factor to consider during deployment is SSL certification. By default, Django apps are deployed on port 80, also known as HTTP. This is fine when hosting the application on the local network. However, HTTPS (usually port 443) is often required for security when deploying applications to the internet. This requires an SSL certificate, which can be obtained through organizations such as Let's Encrypt. However, one can also create a "self-signed" certificate (refer to this [article](#) for more information).

---

## CONCLUSION

Pythonify is a web-based application developed to address the need for a centralized platform for students to view practice problems and track their progress. The application was developed with Django, which has proven to be an efficient framework for database driven web applications. Some key features include:

1. Authentication system for both Students and Teachers.
2. Vibrant dashboards to highlight key information.
3. Ability to filter through practice sets and view coding challenges.
4. Student feedback forms to track progress.
5. Ability to create practice sets and problems.
6. Manage features such as account approval & access code generation.
7. User friendly UI, courtesy of Bootstrap 4.

### Scope for Future Development

The application is highly scalable and flexible for future development. Additional modules and scripts can be built using Python to expand the application's features. Various other database engines such as MySQL, PostgreSQL, and more can be used to replace the current SQLite. Moreover, setting up email notifications with Django's SMTP server support will prove useful for the student authentication process (Forgot Password, User account status, etc).

The source code for this project can be found [here](#).