50.053 Software Testing and Verification

Fuzzing a Django Web Application.

Start with `Docker`

Manual Build

> Download the code

```
$ unzip DjangoWebApplication.zip
```

👉 Set Up for `Unix`, `MacOS`

> Install modules via `VENV`

```
$ virtualenv env #optional
$ source env/bin/activate #optional
$ pip3 install -r requirements.txt
```

> Set Up Database

```
$ python3 manage.py makemigrations
$ python3 manage.py migrate
```

PROF

> Generate API

```
$ python3 manage.py generate-api -f
```

> Start the APP

```
$ python3 manage.py runserver       # start the project
```

At this point, the app runs at `http://127.0.0.1:8000/`.

👉 Set Up for `Windows`

Install modules via `VENV` (windows)

```
$ virtualenv env
$ .\env\Scripts\activate
$ pip3 install -r requirements.txt
```

Set Up Database

```
$ python3 manage.py makemigrations
$ python3 manage.py migrate
```

Start the APP

```
$ python3 manage.py runserver       # start the project
```

At this point, the app runs at `http://127.0.0.1:8000/`.

PROF

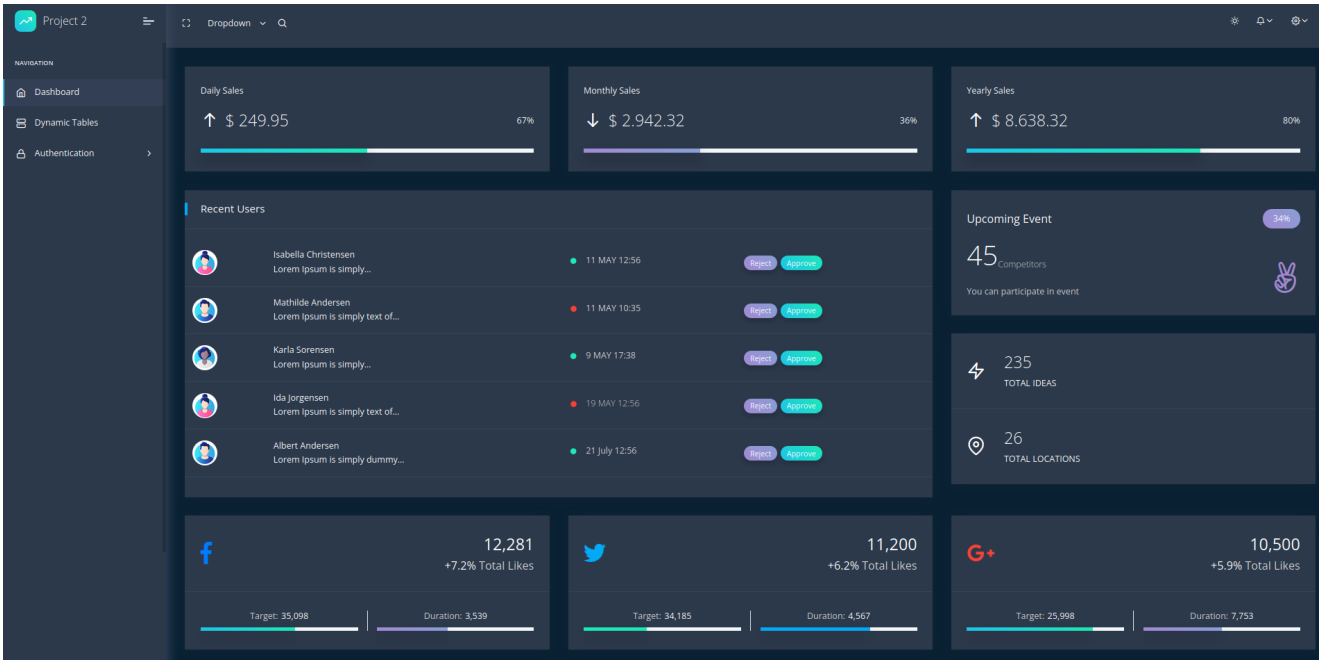

Figure 1. Web application.

Codebase Structure

The project is coded using a simple and intuitive structure presented below:

```
< PROJECT ROOT >
   |
   |-- core/
   |    |-- settings.py                  # Project Configuration
   |    |-- urls.py                      # Project Routing
   |
   |-- home/
   |    |-- views.py                     # APP Views
   |    |-- urls.py                      # APP Routing
   |    |-- models.py                    # APP Models
   |    |-- tests.py                     # Tests
   |    |-- templates/                   # Theme Customisation
   |         |-- pages                   #
   |              |-- custom-index.py     # Custom Dashboard
   |
   |-- requirements.txt                 # Project Dependencies
   |
   |-- env.sample                       # ENV Configuration (default
values)
   |-- manage.py                        # Start the app - Django
default start script
   |
   |--
*******************************************************************************
```

```
# This exists in ENV: LIB/admin_datta
< UI_LIBRARY_ROOT >
   |
   |-- templates/                    # Root Templates Folder
   |    |
   |    |-- accounts/
   |    |    |-- auth-signin.html     # Sign IN Page
   |    |    |-- auth-signup.html     # Sign UP Page
   |    |
   |    |-- includes/
   |    |    |-- footer.html          # Footer component
   |    |    |-- sidebar.html         # Sidebar component
   |    |    |-- navigation.html      # Navigation Bar
   |    |    |-- scripts.html         # Scripts Component
   |    |
   |    |-- layouts/
   |    |    |-- base.html            # Masterpage
   |    |    |-- base-auth.html       # Masterpage for Auth Pages
```

```
    |    |
    |    |-- pages/
    |         |-- index.html           # Dashboard Page
    |         |-- profile.html         # Profile Page
    |         |-- *.html               # All other pages
    |
    |--
**********************************************************************
```

Sending requests

We provide an example of how you can send a `POST` request to add a new item to the database. In this case located at `datatb/product` with the endpoint url `add/`. It is worthy to note that the cookies are optional to include. In this simple example we create a random `name`, `info` and `price`.

```
import requests
import random
import json

base_url = 'http://127.0.0.1:8000/datatb/product/'

endpoint_url = 'add/'

url = base_url + endpoint_url

random_name =
''.join(random.choices('abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUV
WXYZ', k=10))
random_info =
''.join(random.choices('abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUV
WXYZ', k=10))
random_price = str(random.randint(1, 100))

form_data = {
    'name': random_name,
    'info': random_info,
    'price': random_price
}

headers = {
    'Cookie': 'csrftoken=5vvs6151ScRQGpdMlKAf8FAFERO67MmK;
sessionid=c35o5m7xkymbjdtcu9k916f8jfj2f8x7', # Optional
}

try:
    print(json.dumps(form_data))
    response = requests.post(url, headers=headers,
data=json.dumps(form_data))
```

```
    if response.status_code == 200:
        print("Request successful!")
        print("Response:")
        print(response.text)

    else:
        print(f"Request failed with status code:
{response.status_code}")
except requests.exceptions.RequestException as e:
    print("Request failed:", e)
```