Introduction

Purpose of the Guide
The purpose of this administration guide is to provide instructions and guidance for managing and configuring the SAB Trainee Management Application. This guide is intended for system administrators at SAB or other individuals responsible for maintaining and managing the application. The guide aims to provide an overview of the application's architecture and features, as well as detailed instructions for performing common administration tasks such as managing user accounts, configuring system settings, managing content, and ensuring security. The guide also includes troubleshooting steps and answers to frequently asked questions, to help administrators quickly resolve issues and minimize downtime. Overall, the guide serves as a comprehensive reference for administrators to effectively manage and maintain the SAB trainee management web application.

Getting Started

System Requirements
Before installing the web application, ensure that your system meets the following requirements:

- Operating System: Windows, macOS, or Linux (Ubuntu, Debian, CentOS, or Red Hat Enterprise Linux)
- Python: Version 3.6 or later (recommended version: 3.9)
- Django: Version 3.1 or later (recommended version: 3.2)
- Web Server: Apache 2.4 or later, Nginx 1.16 or later, or Microsoft IIS 10 or later (optional, but recommended for production deployments)
- Database: SQLite 3.8.3 or later (recommended for development and testing environments, not recommended for production deployments with high traffic or concurrent users)
- Disk Space: Minimum of 500 MB of free disk space
- RAM: Minimum of 2 GB of RAM

Note that these are minimum requirements, and for optimal performance, we recommend that you exceed these specifications whenever possible. Keep in mind that the system requirements may vary depending on the size and complexity of your installation, as well as the number of concurrent users.

Installation Instructions
To install the web application, follow these steps:

1. Clone or download the latest version of the web application from the GitHub repository to your local machine or server.
   https://github.com/SAB-QA-Team/SAB-Training-Management-App

2. Install the required dependencies, inclduing Python, Django, and a source-code editor (VS-code is recommended)
3. Set up a server instance by using a preconfigured server image, or you can set up a server manually using SSH.
4. Configure the server: Next, you'll need to configure the server to run your Django application. This includes setting up the web server, such as Apache or Nginx, and configuring the appropriate settings file for your Django project
5. Test the application by accessing it locally in a web browser and performing basic tasks such as creating courses, adding trainees, and verifying that the application functions as expected.
6. Deploy your Django application: After configuring the server, you can deploy your Django application to the server. This can be done using a tool like Git, which allows you to push your code to the server and deploy it automatically.
7. Test the application: Finally, you should test your application to ensure that it is working properly. You can do this by accessing the application's URL in a web browser.

Login instructions

1. Open your web browser and go to the URL of the Django application.
2. On the homepage of the application, locate the "Login" button and click it.
3. You will be redirected to the login page, which will prompt you to enter your username and password.
4. Enter your username and password in the appropriate fields.
5. Once you have entered your login credentials, click the "Login" button.
6. If your login credentials are correct, you will be redirected to the application dashboard or homepage.
7. If your login credentials are incorrect, you will see an error message. Please verify that you have entered the correct username and password and try again.

Overview of the Web Application

Architecture

Deployment Environment:
Currently the application is pushed on GitHub, not connected to any server. Once the new owners of the application receive they can host it as that was their preferred approach.

Backend technologies:
The web application was implemented using Python with Django, as it fits the requirements we had for implementing the application. Features like: authentication, authorization, admin terminal, otp library (2FA feature), etc. were all factors that gave Django the upperhand for being the ideal option for our project.

Data storage:
For the purpose of this project we used the default database in django apps (sqlite 3), but the new owner easily convert the database into the type they seek by just installing the database package on python and modifying the settings.py of the application.

Configuration and Customization:
Several aspects of the were customized to fit the scope of our project. Starting with users, Django apps have their own default users, but for the purpose of this project custom users were created to fit the standards of SAB. Additionally, authorizations like 2FA using Google Authenticator are implemented within the application to satisfy security requirements. One important feature is that all web pages are available in both English and Arabic.

Main features
The SAB trainee management web application is intended to provide the managers of external state entities the ability to efficiently keep track of their employees' training progress, as well as to provide trainees with the ability to efficiently and effectively manage their training courses. Our web application will include the following features:

1. See recommended courses
   Our solution will provide trainees with recommended courses based on their profile, expertise, and current job position. Initially, we plan to recommend courses based on the various fields (Accounting, Auditing, IT, etc) . If feasible, we might implement a model that processes trainees information and recommend courses based on the input data.
2. Sign up for a course
   The system will allow the trainees to sign up for different courses. Once they sign up, our system will automatically send emails to the trainees and their managers regarding the timing, material, and logistics of the course.
3. Record attendance
   Our system will also allow the trainee to record their attendance for each class through the web application, by inputting a unique code that will be revealed during the class, thereby keeping track of a trainee’s progress.
4. Receive certificates
   The system will automatically send certificates to the trainees once the 90% course completion requirement has been met.
5. Track progress
   Our system will also allow managers of different state entities to track and check the training progress of their employees.

User roles and permissions
Trainee

- Sign up and login

- See the list of courses and thier details
- Register for a course
- Track thier attendance for a course

Managers
- Login
- View a dashboard of summary statistics
- Track their employees' courses and attendance

Admin
- Login
- See the list of courses, trainees, managers, and instructors
- Adding, removing, and edit the courses

Administration Tasks

Managing entities
The SAB trainee management application allows the admins of SAB to manage the all the entities within the application through the Django admin interface. This system enables administrators to create, edit, and delete instances of different classes easily and efficiently

To manage the entities:
1. Launch the application through a web browser
2. Type /admin at the end of the URL
3. Login in with the following credentials:
   Username: …..
   Password: …..
4. Select the class you want to handle
5. Select the functionality you want to use (add, update, delete)

Configuring Systems Settings
To configure system settings for a Django application, follow these steps:
1. Open the settings.py file in a text editor.
2. Locate the desired settings section and modify the settings as necessary.
3. Save the settings.py file and restart the Django application to apply the changes.

By configuring system settings for a Django application, developers can customize the behavior of the application to meet the needs of their organization and ensure that it operates effectively and securely.

Managing Content
To manage the content of the application, the admin needs to access the files depending on the change needed. Below is the list of content elements and their corrosponding files

To manage the database entities or attributes:

1. First, navigate to the models.py file in your Django application's directory. The models.py file contains the classes that define the structure of your application's database.
2. In the models.py file, you can define a class for each of the objects that you want to store in your database. Inside the class, you can define the fields that you want to store using Django's field classes. You can also edit he existing classes and their fields
3. Once you have defined/modified your classes in the models.py file, you need to create the corresponding database tables using Django's migration system. To do this, run the following commands in your terminal:
    - Python manage.py makemigrations
    - Python manage.py migrate

To mange the functionalities of the application:

1. First, navigate to the view.py file in your Django application's directory. The view.py file contains the functions that are responsible for handling requests and rendering responses for your application.
2. In the view.py file, you can define functions for each of the functionalities that you want to implement in your application. You can also edit the content of existing functions.
3. Once you have added your new functions or modified the existing functions, save the file

To manage the routes of the application:

1. First, navigate to the urls.py file in your Django application's directory. The urls.py file contains the URL patterns for your application.
2. In the urls.py file, you can define a URL pattern using Django's url function. The url function takes two arguments: a regular expression that matches the URL, and a view function that handles the request. You can also use named URL patterns to make it easier to refer to URLs in your templates and other code. To define a named URL pattern, use the "name" argument of the url function.
3. Once you have added new URLs or modified existing ones, save the file

Testing
Testing is an essential part of the development process that ensures the quality and reliability of your web application. In this section, we'll go over the steps to set up a virtual environment and install the necessary packages to run your application's tests.

1. Install virtualenvwrapper

Virtualenvwrapper is a Python package that helps create and manage virtual environments. A virtual environment is a self-contained environment that allows you to install packages and dependencies without affecting the global Python installation on your system.

To install virtualenvwrapper, use the following command in your terminal:

```
pip install virtualenvwrapper
```

2. Create a virtual environment for testin

Once you have installed virtualenvwrapper, create a new virtual environment for testing using the following command:

```
mkvirtualenv testing
```

This command will create a new virtual environment named 'testing' and activate it.

3. Activate the testing virtual environment

To activate the testing virtual environment, use the following command:

```
workon testing
```

This command will activate the 'testing' virtual environment, and you should see the virtual environment's name appear in your terminal prompt.

4. Save the dependencie

Next, save the dependencies required for your testing environment using pip freeze. This command will output a list of all the packages and their versions installed in your virtual environment:

```
pip freeze > requirements.txt
```

5. Install the dependencies

To install the dependencies saved in the requirements.txt file, use the following command:

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

This command will install all the dependencies necessary to run your web application's tests.

Now that you have set up your virtual environment and installed the necessary packages, you're ready to run tests for your Django web application. You can use the Django testing framework to write and run tests for your web application.

To access the curent tests and write new ones, navigate to the ‘test’ folder
You can run all tests through the following command:

```
python manage.py test
```