Project Idea

The idea behind the project is to create an entry management system that can accept Visitor and host as the account type, stores the data of both users and provide visitor user with accessibility to schedule meeting at any time on a day.

Project Description

This project is developed with a web-based-development approach and it uses an SQLite database to store user data and Bootstrap for frontend. It is completely developed on the Django framework and it's built with function-based-views.

The first screen comes with the home template that allows a user to signup or login. Once the user signs up, all the fields get stored into the database, invalid entry of details displays corresponding errors. After successful creation of an account, an email is sent to registered email id with all the registered details and user is directed to details template where their details are portrayed. The login window also loads the details template upon successful authentication of a user.

Once a user gets logged in, their views are different for two types of users namely(visitor, host). A visitor is allowed to schedule the meeting whereas host doesn't hold this functionality. Upon clicking of meeting link, the visitor is directed to meeting template where he can schedule a meeting with a host. The host list(hostname, host id) is displayed on the same template to allow a visitor to enter corresponding host id along with other details include check-in-time, check-out-time, address. Upon successful scheduling of meeting, email and SMS is sent to respective host id, chosen by the visitor during scheduling of meeting regarding the details of the meeting and visitor is directed back to details template. Immediately past the check-out-time provided by the visitor during scheduling a meeting, an email is sent to the visitor regarding the details of the meeting. All meetings scheduled by any visitor also gets stored into the database along with a timestamp.

Views Highlights

- Signup View: This view contains all the code to signup, validate and then authenticate a user. Based on the received details from user form it first validates form data then creates the model object to create an account in the database, upon successful creation of an account, it proceeds to send email to the registered user.

- Login View: This view contains all the code to validate an already registered user. It authenticates the user and redirects them to the details template.

- Logout View: This view logout user on clicking logout link on the webpage and it redirects the user to the home template.

- Meeting View: This view contains all the code to let the user gets to the meeting page template, displays all the host available in the database, creates a model object upon successful validation of meeting form, sends email and SMS to the chosen host just after successful scheduling of meeting and schedule the job to send the email to visitor past the check-out-time of the meeting. It redirects visitor the detail template upon successful meeting event other error gets raised.

- Relogin View: It just re-login the pre logged in user upon any update in the database of the user whenever required.

- Send Email View: It contains all the code to send an email to the user.

- Send SMS View: It contains all the code to send an SMS to the user.

Apps Highlights

- Account App: It contains all the views,models, forms, tasks and urls of the project. Two models used are Account which contains Name, Email, Password, Phone, Account type fields and another Meeting which contains HostID, check-in-time, check-out-time, address, timestamp.

- Ems App: It contains settings of project and base templates used in the project

Workflow Description

The workflow has two main apps: 'accounts' and 'ems'.

Account app contains all the major functional code of the project and the ems app contains settings of the project.

Requirements.txt contains all the libraries used in the project along with the version. 'manage.py ' is the root of the project.

Special Libraries Used

smtplib: To send mail.

Twilio: To send SMS.

Celery: To schedule mail and message after a time interval. Task Broker used for pushing the task is Redis server.

Note:

- All the usernames, passwords and keys are '*' marked. Comments are written right next to statement to guide for what is required in the field.
- This project is done in a virtual environment. It is highly recommended to run this project in a virtual environment.
- Default 'User ' model is altered to store email address as the username of the user.