



Sindibad[®]

SINDIBAD TOUR GUIDE SEARCH ENGINE

Software engineering documentation



NANKAI University
College of Software
Web Application Development
Spring 2023

ID	Name
2120226034	SAID SAIF MOHAMMED ALSAIDI
2120226096	MOHAMMED SUAD ALI
2120226097	AMMAR KHALID MOHAMMED
2120226098	HIDER HUSAM AHMED

Table of Contents

Introduction:	2
Project Overview:	2
Objectives:	3
Project purpose:	3
Scope:	3
Constraints:	3
Team members:	4
Relative technologies analysis:	6
System Analysis:	12
Functional requirements:	13
Non-Functional requirements:	14
System Design:	14
Presentation Layer:	15
Data design:	19
Entity relationship diagram (ERD):	19
Class diagram:	20
Use case diagram (UCD):	22
Data flow diagram (DFD):	23
Activity diagram:	25
Implementation:	26
Testing phase:	39
FooTest:	40
CI tool:	41
Deployment:	43
Uploading Django Code to Hosting Server:	45
Create VirtualEnv and Install Django and Dependencies:	46
Setting up Django Web app and WSGI file:	47
Create a Web Application with Manual Config:	47
Edit WSGI File to Point our Django Project to Server:	48
Create MySQL database for our website:	49
Conclusion:	50

Introduction:

When a tourist visits a country for any purpose whatsoever, the most important thing he is looking for is basic amenities such as a hotel, restaurant, entertainment, etc. Today, companies through digital technology provide many services and facilities for tourists to book and choose all these things through mobile applications or websites. The schedule of trips for tourists through these applications is a common culture for most people, but the large number of applications has become annoying to many people, and the presence of an application or website that collects all the needs of tourists is a unique idea that will solve many problems and save time for many tourists.

Project Overview:

Sindibad is a search engine for the tourist through which he can search for the city or country he wants to visit or is currently visiting; it is derived from the name of the famous Arab traveler called Sindibad. This search engine is to find out hotels, restaurants, entertainment venues and tour guide companies, as well as tourist can see their details. This engine can specify searching in all cities within the country or in a specific city and to see the details. Also, this engine is linked to a special control panel that allows the administrator to register data related to the search results, such as registering countries, cities, amenities, and entertainment that the tourist is looking for. It will also be available for hotel companies, restaurants, entertainment, and tourist guide companies to register their services and provide an opportunity for everyone to enrich the content of the site with all that the tourist needs, with the possibility of developing it in the future to include customer evaluation of the services provided, as well as booking through the site.

Objectives:

- Improve the quality of tourism services.
- Help tourists to get the best experience through the website.
- Enhance the role of technology to provide good solutions for the current situations.
- Saving time and effort for users.
- Enhancing the quality of tourism sectors in the countries.

Project purpose:

The purpose of this project is to provide a search engine for tourists to facilitate their tourism experience and gather everything they need in one platform that brings together all needs in one place and one digital platform.

Scope:

The app will help tourists and any users to get what they need to get best tour in the cities, instead of using many apps and consume time of registering and accessing all these platforms to check hotels, restaurants, entertainments and tour guide companies. The app will provide many features to make users satisfied with their experience, with take care of the quality of services that the app provides and the integrity of information. Tourism will take advantage of promoting the services in the countries via this app.

Constraints:

When tourists visit new countries, the most important thing for them is to find comfortable hotel and place to eat and other entertainments to get best experience, there are many popular apps that can help tourists to get what they need, many companies provide what tourists need but the tourist must register in

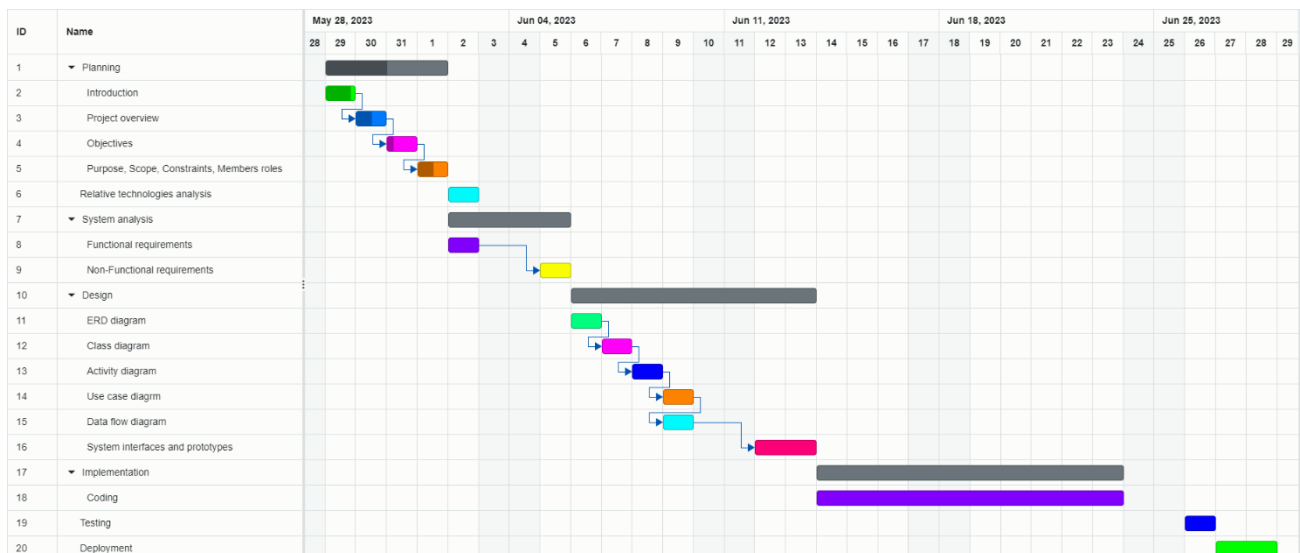
every app and search in many platforms to get what he needs. Sindibad tour guide is a website to help people everywhere search for the most popular places in the countries and cities to visit and get good experience and discover the cities through one platform collect all these services in one place.

Team members:

#	Name	ID	Tasks
1	SAID SAIF MOHAMMED AL SAIDI	2120226034	Project manager: <ul style="list-style-type: none"> • Prepare the work environment in GitHub. • Distribute the work among team members. • Create database. • Implement coding of cPanel in (MVC). • Create & design CSS3 file. • Testing and deployment. • Write documentation including (introduction, part of planning, relative technologies, analysis phase, part of implementation phase, deployment, conclusion).
2	MOHAMMED SUAD ALI	2120226096	Team member: <ul style="list-style-type: none"> • Implement home page for visitors with all functionalities. • Work on analysis phase. • Write documentation including (part of Design phase, part of planning).
3	AMMAR	2120226097	Team member:

	KHALID MOHAMMED		<ul style="list-style-type: none"> Implement login page with all functionalities & logout. Work on analysis and design phase, diagrams, and some functionalities of testing phase.
4	HIDER HUSAM AHMED	2120226098	<p>Team member:</p> <ul style="list-style-type: none"> Write part of planning phase and help in deployment phase documentation. Implement CRUD operations of users and roles. Help in designing the interface of the website.

The work is distributed among group members with specified schedule, all the members participated in analysis, design and implementation phase, GitHub tool will be used to arrange each coding task among the team members. The Gannt chart below describes the schedule for each task.



Relative technologies analysis:

The project is a website, the main programming language will be used is Python in the server-side section with a MySQL database for programming and designing the system's database tables. Regarding the front-end side, HTML5 will be used with CSS3 to design the website interface, in addition to JavaScript to control some features, we will not use any theme like bootstrap in our project, we will code everything from scratch.

The Django framework is used in design and programming the website. It is one of the most popular web design techniques in the Python language because it contains great features that organize the work for programmers, in addition to the presence of many software libraries available within it, as this framework relies on the MTV technology in programming, where M represents the models and the T templates, which including the user interfaces and some other files related to the design of the website, while the V is specific to the views file, which is a controller that acts as a bridge between the models and the templates.

Django is a popular Python web framework that follows the Model-View-Controller (MVC) architectural pattern but as MTV as mentioned above, where views.py is acted as controller and templates as views. It provides a set of tools and libraries to help developers build robust and scalable web applications quickly and efficiently.

Here are some key features and concepts of the Django framework:

- Object-Relational Mapping (ORM): Django provides a high-level ORM that allows you to interact with your database using Python code instead of

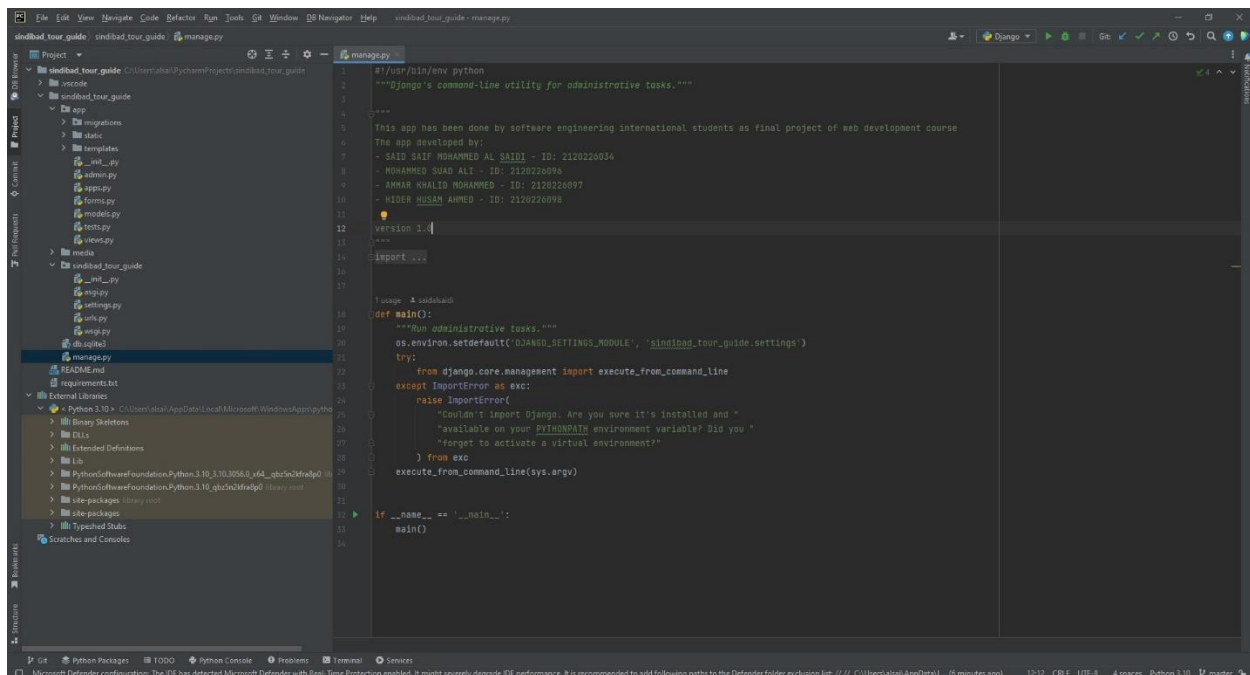
writing raw SQL queries. It supports multiple database backends and simplifies database operations.

- URL routing: Django uses a URL routing system to map incoming HTTP requests to the corresponding view functions. You can define URL patterns in a configuration file, which helps in building clean and user-friendly URLs.
- Template system: Django has a powerful template system that separates the presentation logic from the business logic. Templates are HTML files with special syntax that allows you to dynamically generate content and display data from the backend.
- Forms handling: Django provides a forms library that simplifies the handling and validation of user input. It helps in generating HTML forms, processing form submissions, and performing validation on user data.
- Authentication and authorization: Django includes a built-in authentication system that handles user registration, login, logout, and password management. It also provides an authorization framework to manage user permissions and access control.
- Admin interface: Django offers an automatic admin interface that allows you to manage your application's data models without writing any additional code. It provides an easy-to-use interface for performing CRUD (Create, Read, Update, Delete) operations on your database records.
- Security features: Django incorporates various security features to protect your web applications from common vulnerabilities, such as cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection.
- Scalability and extensibility: Django is designed to handle high-traffic websites and can scale horizontally by employing techniques like load

balancing and caching. It also has a robust ecosystem of third-party packages and extensions, making it highly extensible.

Django follows the "Don't Repeat Yourself" (DRY) principle, emphasizing code reusability and reducing redundancy. It promotes clean and maintainable code, making it a popular choice for building complex web applications.

The used IDE to develop the website is PyCharm (community edition 2023.1.2), it is an integrated development environment (IDE) specifically designed for Python programming. It provides a wide range of features and tools to help developers write, debug, and test Python code more efficiently. It's very popular IDE and support python language, it is a lightweight, and highly extensible source code editor developed by Microsoft. It is available for Windows, macOS, and Linux and provides a rich set of features that make it popular among developers.



Here are some key features of PyCharm:

- Code Editor: PyCharm offers a powerful code editor with syntax highlighting, code completion, and code formatting features. It helps improve code readability and productivity.
- Intelligent Code Completion: PyCharm's intelligent code completion suggests code snippets, method names, and variable names as you type, saving time and reducing the chances of making syntax errors.
- Code Navigation and Refactoring: PyCharm provides advanced code navigation features like Go to Definition, Find Usages, and Code Structure View. It also supports automated code refactoring, making it easier to restructure and improve the quality of your code.
- Debugging and Testing: PyCharm offers a built-in debugger that allows you to step through your code, set breakpoints, inspect variables, and track the flow of execution. It also supports unit testing frameworks like unittest, pytest, and doctest, making it easy to write and run tests.
- Version Control Integration: PyCharm seamlessly integrates with popular version control systems like Git, Mercurial, and Subversion. It provides features for committing, updating, merging, and managing code repositories directly from the IDE.
- Virtual Environments and Package Management: PyCharm supports creating and managing virtual environments, allowing you to isolate project dependencies. It also provides integration with package managers like pip and conda, making it easy to install, update, and manage Python packages.
- Web Development Support: PyCharm includes features specifically tailored for web development using frameworks like Django, Flask, and Pyramid. It provides support for templates, URL routing, database integration, and more.

- **Integration with Other Tools:** PyCharm integrates with popular tools like Jupyter Notebook, Anaconda, Docker, and SQL databases. It offers seamless integration, enabling you to work with these tools within the IDE.
- **Extensibility:** PyCharm is highly customizable and extensible. You can install and configure various plugins and extensions to enhance its functionality and tailor it to your specific needs.

Overall, PyCharm is a robust and feature-rich IDE that helps Python developers streamline their workflow, improve code quality, and increase productivity.

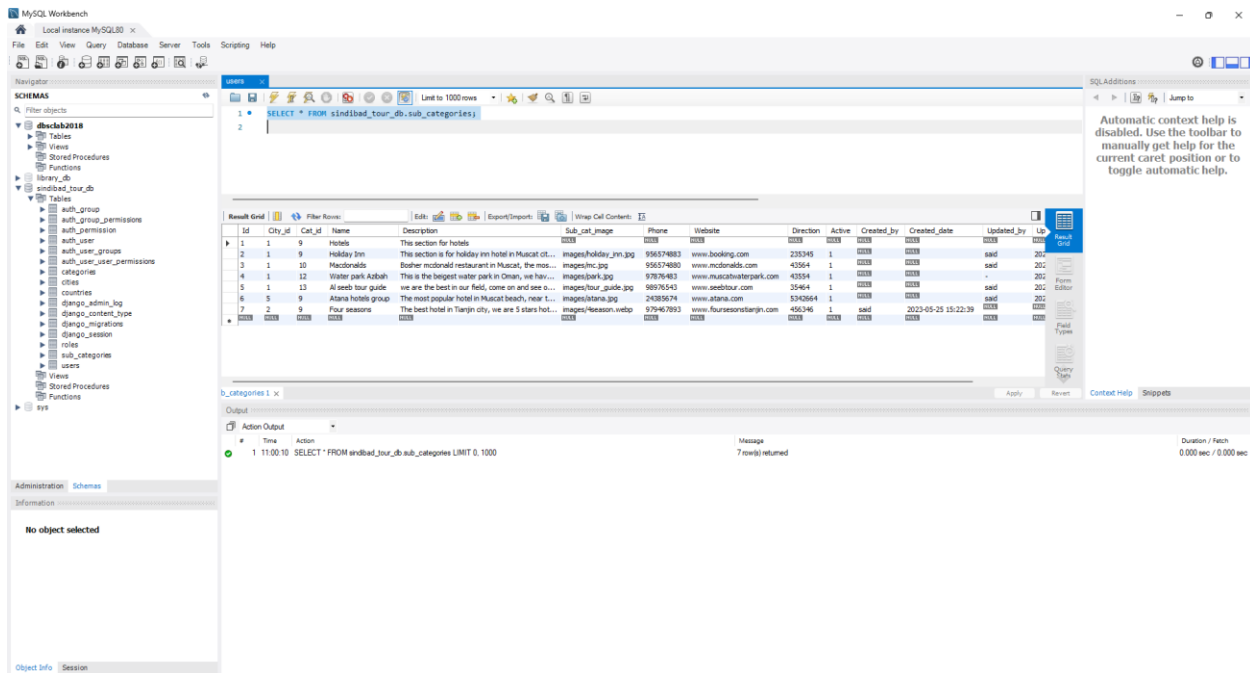
The type of database that is used is relational database management system, will be implemented in MySQL workbench, it is a popular graphical tool for managing and interacting with MySQL databases. MySQL Workbench has many features make it different of other DBMS such as:

- **Visual database design:** MySQL Workbench provides a visual interface for designing and modeling databases. It allows you to create and modify database schemas using a drag-and-drop interface, making it easier to design and understand the structure of your database.
- **Query development and execution:** MySQL Workbench offers a built-in SQL editor that provides syntax highlighting, code completion, and error checking. You can write and execute SQL queries directly within the tool, making it convenient for database development and testing.
- **Database administration:** MySQL Workbench provides a wide range of administrative features for managing MySQL databases. It allows you to create, alter, and drop databases, tables, and indexes. You can also manage user accounts, set privileges, and monitor server status and performance.

- Data migration and synchronization: MySQL Workbench supports data migration and synchronization between different database instances. You can easily transfer data from one MySQL server to another, even if they are on different platforms or versions. This feature is useful when migrating databases or keeping multiple database instances in sync.
- Performance tuning and optimization: MySQL Workbench includes tools for performance analysis and optimization. It provides visual representations of query execution plans, allows you to profile queries, and suggests improvements based on indexes and statistics.
- Database backup and restore: MySQL Workbench allows you to perform full or partial backups of your databases and restore them when needed. It provides options for scheduling backups, configuring backup retention policies, and automating backup tasks.
- Database documentation: MySQL Workbench enables you to generate visual documentation of your database schema, including tables, relationships, and indexes. This documentation can be useful for sharing database designs, understanding the database structure, and collaborating with team members.
- Cross-platform support: MySQL Workbench is available for Windows, macOS, and Linux, making it accessible to developers on different operating systems.
- Integration with other MySQL tools: MySQL Workbench integrates seamlessly with other MySQL tools and utilities. For example, it can connect to MySQL server instances managed by MySQL Enterprise Edition, providing access to additional monitoring and security features.
- Community and support: MySQL Workbench has a large and active user community, which means you can find resources, tutorials, and forums to seek help and share knowledge. Additionally, as an official tool provided by

MySQL, it is well-maintained and regularly updated to support the latest MySQL features and versions.

Overall, MySQL Workbench is a powerful and versatile tool that simplifies database management tasks, enhances productivity, and provides a user-friendly interface for working with MySQL databases.



System Analysis:

The search engine provides functionality to search for various categories relevant to tourists, including hotels, restaurants, entertainment venues, and tour guide companies. Users can search for these categories and view detailed information about each result. The engine allows users to search either across all cities within a country or within a specific city, offering flexibility and personalized search options.

The search engine is also connected to a dedicated control panel, which serves as an administration tool for the system. The administrator can use this

control panel to register and manage data related to the search results. This includes registering countries, cities, amenities, and entertainment options that tourists are looking for. The control panel acts as a central hub for maintaining the database and ensuring accurate and up-to-date information is available to users.

Furthermore, the search engine provides an opportunity for businesses such as hotel companies, restaurants, entertainment venues, and tourist guide companies to register their services. This registration feature allows these businesses to showcase their offerings and contribute to enriching the content of the site with all the necessary information for tourists. The project also mentions the possibility of future development to include customer evaluations of the services provided and enable booking functionality directly through the site.

Functional requirements:

Here is the set of requirements that enable the system to interact with the visitors. The main aim will be to provide this application to the audience which can enable them to achieve successful tour experience through simple search engine to helping them finding the services and attractive places in that region.

- 1- Admin can register new countries in the system.
- 2- Admin can register cities linked with countries.
- 3- Admin can register services categories in the system.
- 4- Admin can register subcategories in the system linked to cities and main categories.
- 5- Admin can edit, update and delete all above points.

6- Visitor or tourist can search via website for all categories with their related information by entering keywords of cities and countries.

Non-Functional requirements:

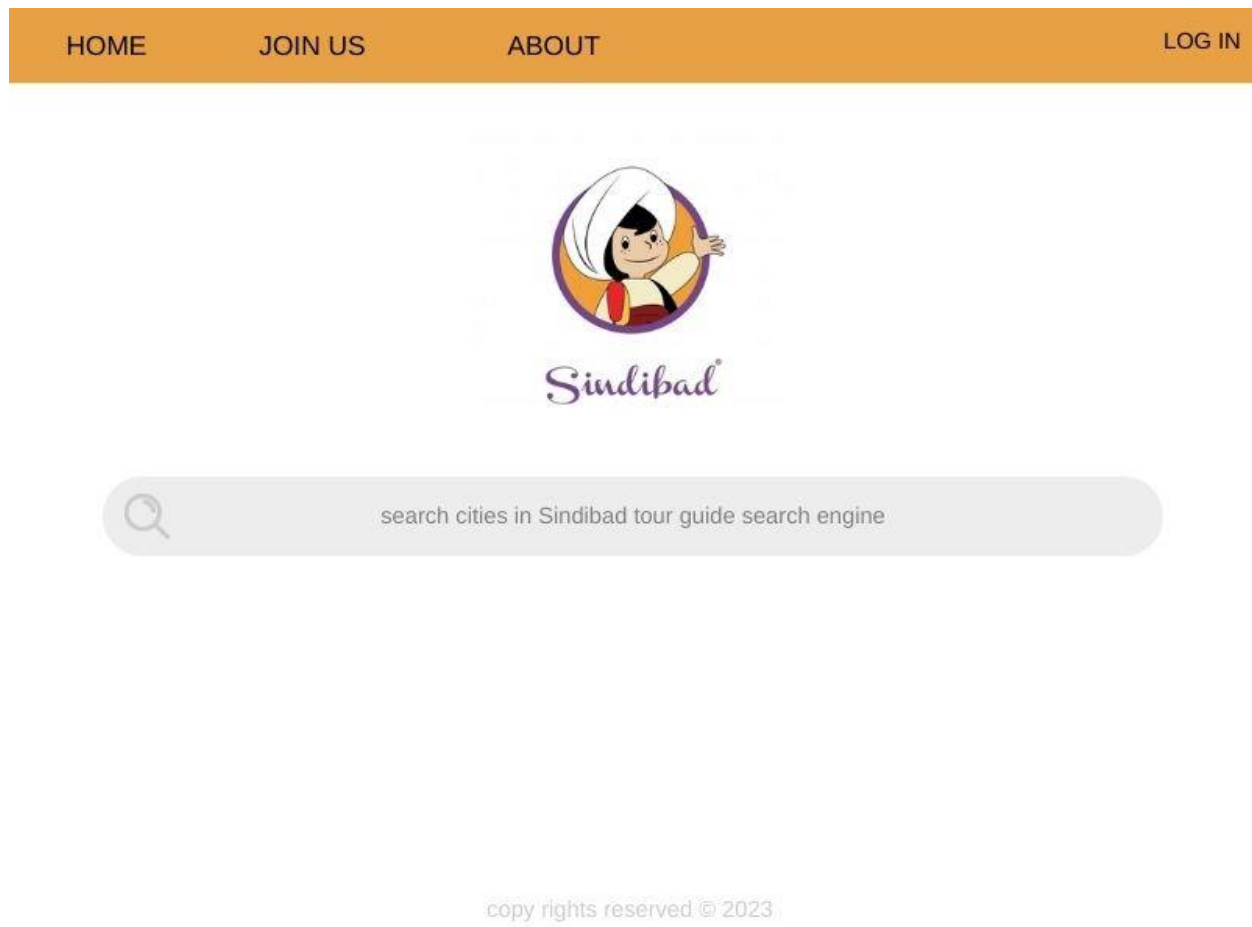
- 1- Security: Django will provide the latest techniques to protect information and validate data via python.
- 2- Update: information may change between time to time, so the information inside the app should be changed according to the new data which we have.
- 3- Integrity: the information should be real and available for the user.
- 4- Usability: the application will provide a simple page for visitors to search for the services, including clear navigation buttons for admin also in control all sections.

System Design:

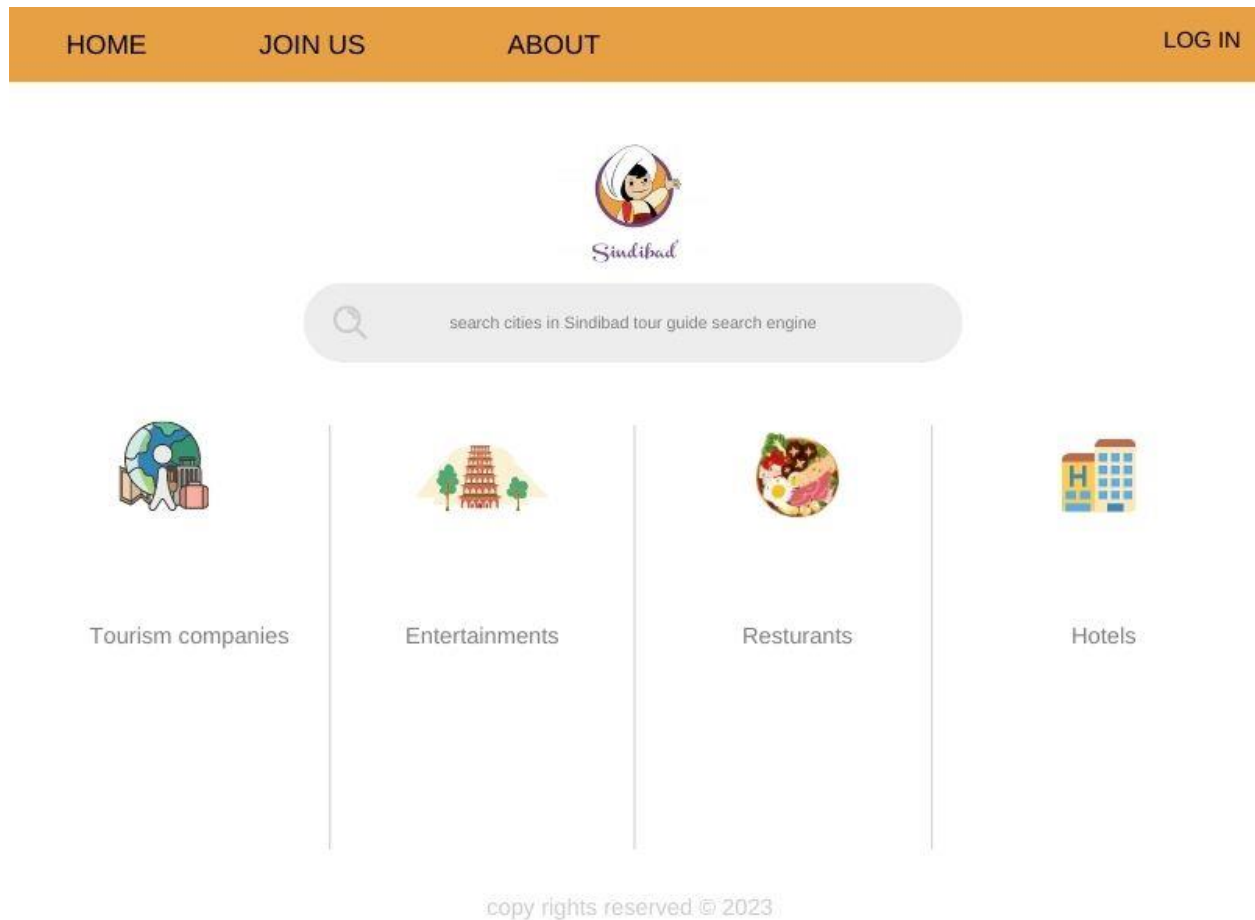
This phase the aim is to translate the requirements gathered in the previous phase into a detailed design that outlines the architecture, components, and interactions of the web application. The web application follows a three-tier architecture consisting of a presentation layer, application layer, and data layer. The presentation layer will handle user interactions, the application layer will process requests and business logic, and the data layer will manage the storage and retrieval of data.

Presentation Layer:


The user interface will be developed using HTML, CSS, and JavaScript. It will include responsive design principles, consistent navigation, and interactive components. Here are the prototypes of the interface designs of the website:



In the previous prototype the visitor interface to search for services such as hotels, restaurants etc. With fixed navbar to let him access other links such as about and join us.



Here are the results when the tourist searches by country name or city name and how the results will show up to him. The app contains complicated algorithms to help the user get the best result when he tries to search for services.



Sindibad



username

password

Log In

The login form prototype, designed to ask admin user for registered username and password to let him log in to control panel and control everything.

HOME	COUNTRIES	CITIES	CATEGORIES	SUBCATEGORIES	USERS	LOG OUT
------	-----------	--------	------------	---------------	-------	---------

COUNTRIES					Create +
#	Name	Description	Icon	Actions	
1	China	This section for China		Edit	Delete View
2	USA	This section for USA		Edit	Delete View

Here is an example of controlling data, especially countries, display countries information in table, each row contains information about specific country with button links such as edit and delete to control information. We can imagine that the control panel contains fixed navbar to control everything

including countries, cities, categories, subcategories and control users of the cPanel.

A form prototype for adding a new country. It features a navigation bar at the top with links: HOME, COUNTRIES, CITIES, CATEGORIES, SUBCATEGORIES, USERS, and LOG OUT. Below the navigation bar, the title 'COUNTRIES' is displayed. The form consists of three input fields: 'Enter name of country', 'Enter Description of country', and 'Add Icon of Country +'. A green 'Save' button is positioned at the bottom of the form.

Form prototype to add country with name, description and image fields.

A form prototype for deleting a country. It features a navigation bar at the top with links: HOME, COUNTRIES, CITIES, CATEGORIES, SUBCATEGORIES, USERS, and LOG OUT. Below the navigation bar, the title 'COUNTRIES' is displayed. The form shows the details of a country to be deleted: 'Name: China' and 'Description: This section for China'. Below the description is an image of the Chinese flag. A red 'Delete' button is positioned at the bottom of the form.

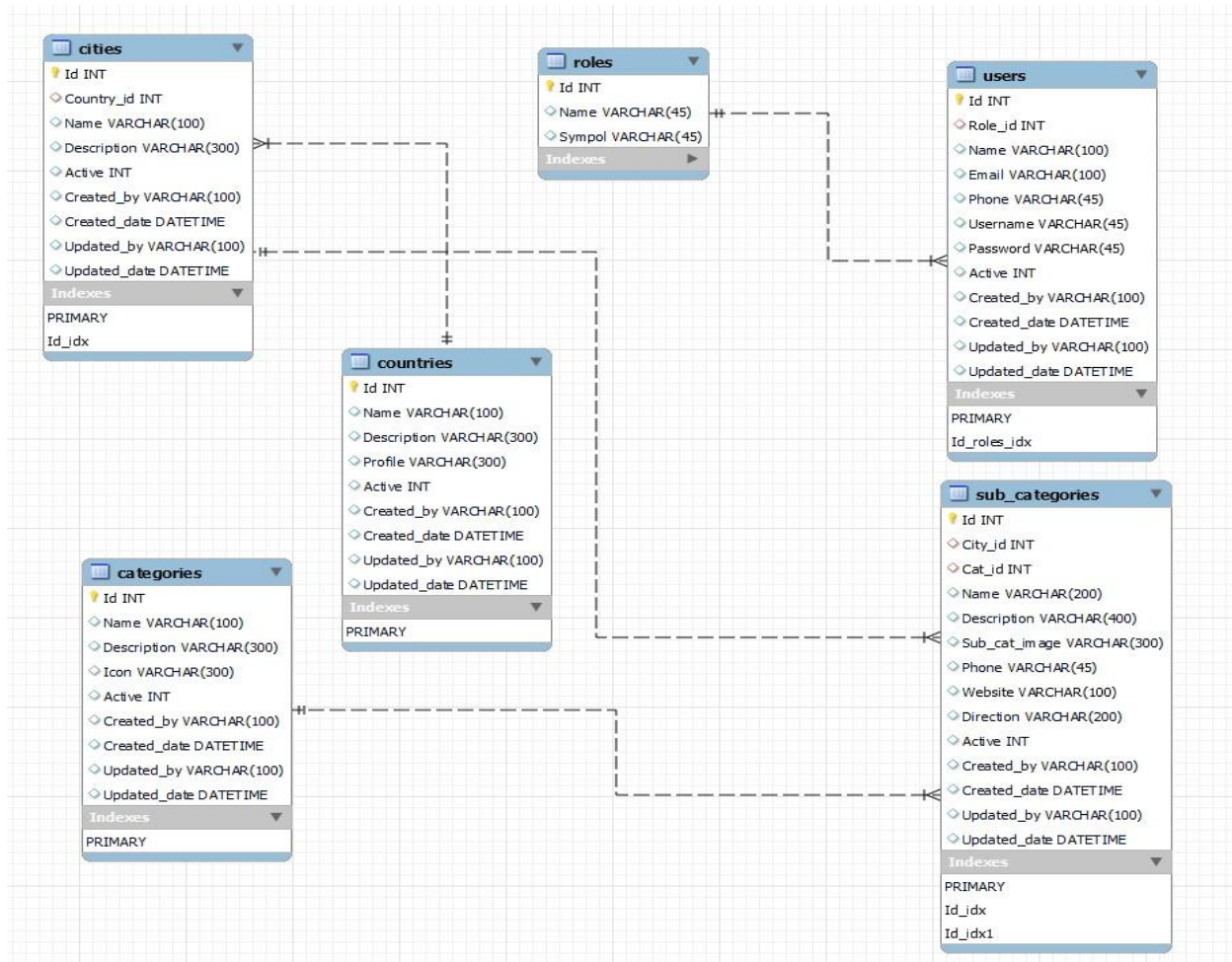
The delete page prototype shows information of each item before deleting, with button to confirm deleting.

All other models contain the same interface prototypes, with different information for each table. The previous prototypes implemented via client-side using HTML5, CSS3 and JavaScript, with Django framework using python as server-side language. For query language, MySQL workbench is the engine to store data using relational database model.

Data design:

Entity relationship diagram (ERD):

System database consists of six tables, countries table to register each country, cities linked to each country, so each country has one city or many, the relationship type between them one to many, and there is also categories table to register type of services such as hotels, restaurants, entertainments and tour guide companies, subcategories table to register information type of each service, linked to its category type and city, so each city can has one subcategory or many as well as each category can has one subcategory or many, the type of relationship one to many. The users table to control administrators who control the cPanel, linked with roles table, each role can have one user or many. The next entity relationship diagram explains all the tables with their attributes and relationships:



So, the cities table contains index foreign key Country_id associated with countries table primary key Id, the sub_categories table contains two indexes foreign keys which are City_id and Cat_id associated to cities and categories tables primary keys, so countries, cities and categories are master tables and sub_categories is details table which has all information about services.

Class diagram:

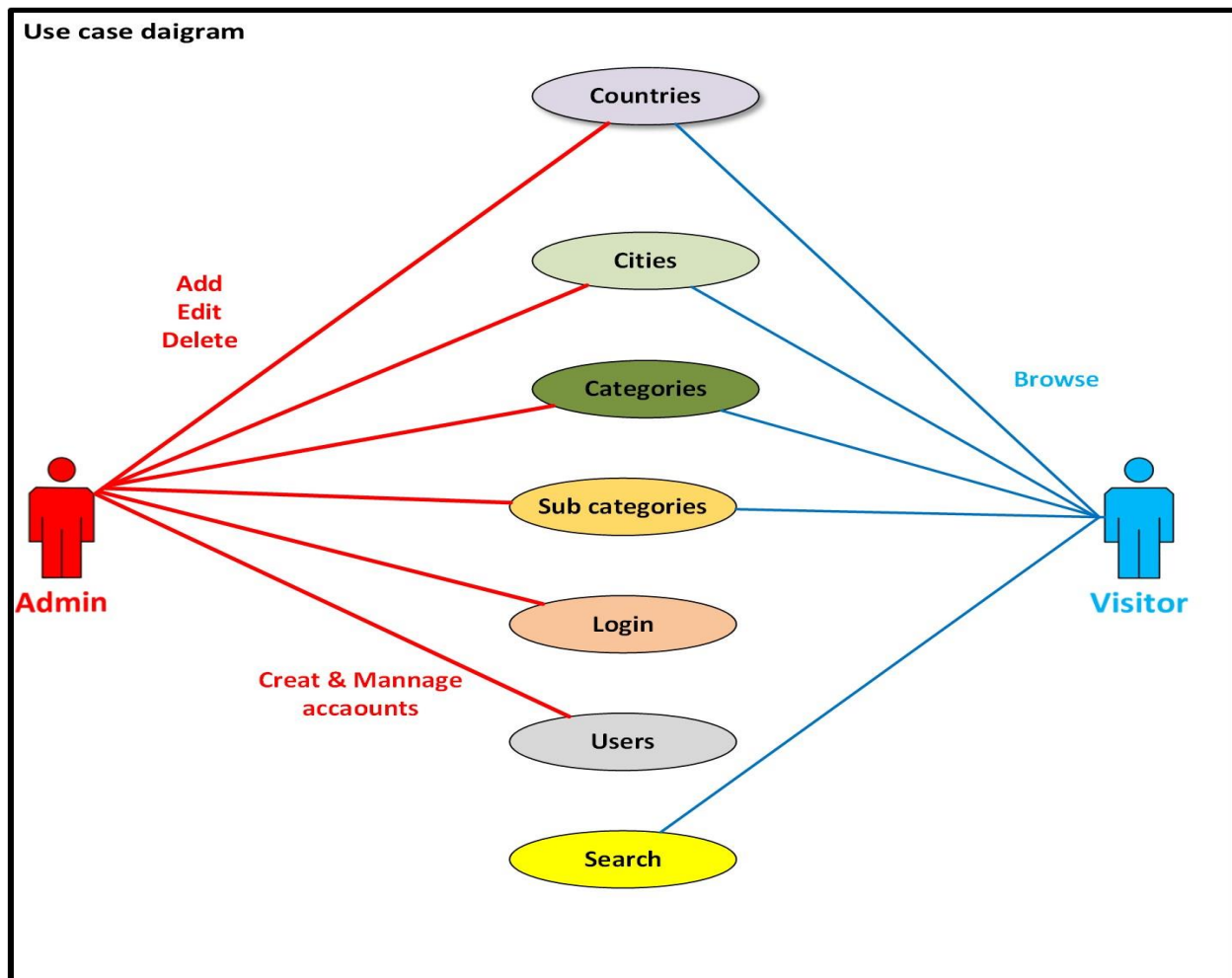
In next class diagram, we have six classes which are Countries, Cities, Categories, Subcategories, Users and roles classes, the Countries class has attributes such as 'Id', 'Name', 'Description', 'Profile', 'Active', 'Created_by', 'Created_date', 'Updated_by' and 'Updated_date', and three defined methods such

as 'AddCountry()', 'UpdateCountry()' and 'DeleteCountry()'. The Cities class represents city and has attributes such as 'Id', 'Country_id', 'Name', 'Description', 'Active', 'Created_by', 'Created_date', 'Updated_by' and 'Updated_date', with there methods 'AddCity()', 'UpdateCity()' and 'DeleteCity()'. The 'Country_id' is attribute in the Cities class is a foreign key that establishes a relationship with the Countries class. The Categories class has same attributes and methods of Countries class, it has attributes such as 'Id', 'Name', 'Description', 'Icon', 'Active', 'Created_by', 'Created_date', 'Updated_by' and 'Updated_date', and three defined methods such as 'AddCategory()', 'UpdateCategory()' and 'DeleteCategory()'. In SubCategories class there are two attributes representing the foreign keys with Cities and Categories, this indicates that the SubCategories class references the 'City_id' attribute in the Cities class and 'Cat_id' attribute in the Categories class. SubCategories class also contains attributes such as 'Id', 'Name', 'Description', 'Phone', 'Website', 'Direction', 'Sub_cat_image', 'Active', 'Created_by', 'Created_date', 'Updated_by' and 'Updated_date', with three methods 'AddSubCategory()', 'UpdateSubCategory()' and 'DeleteSubCategory()'.



Use case diagram (UCD):

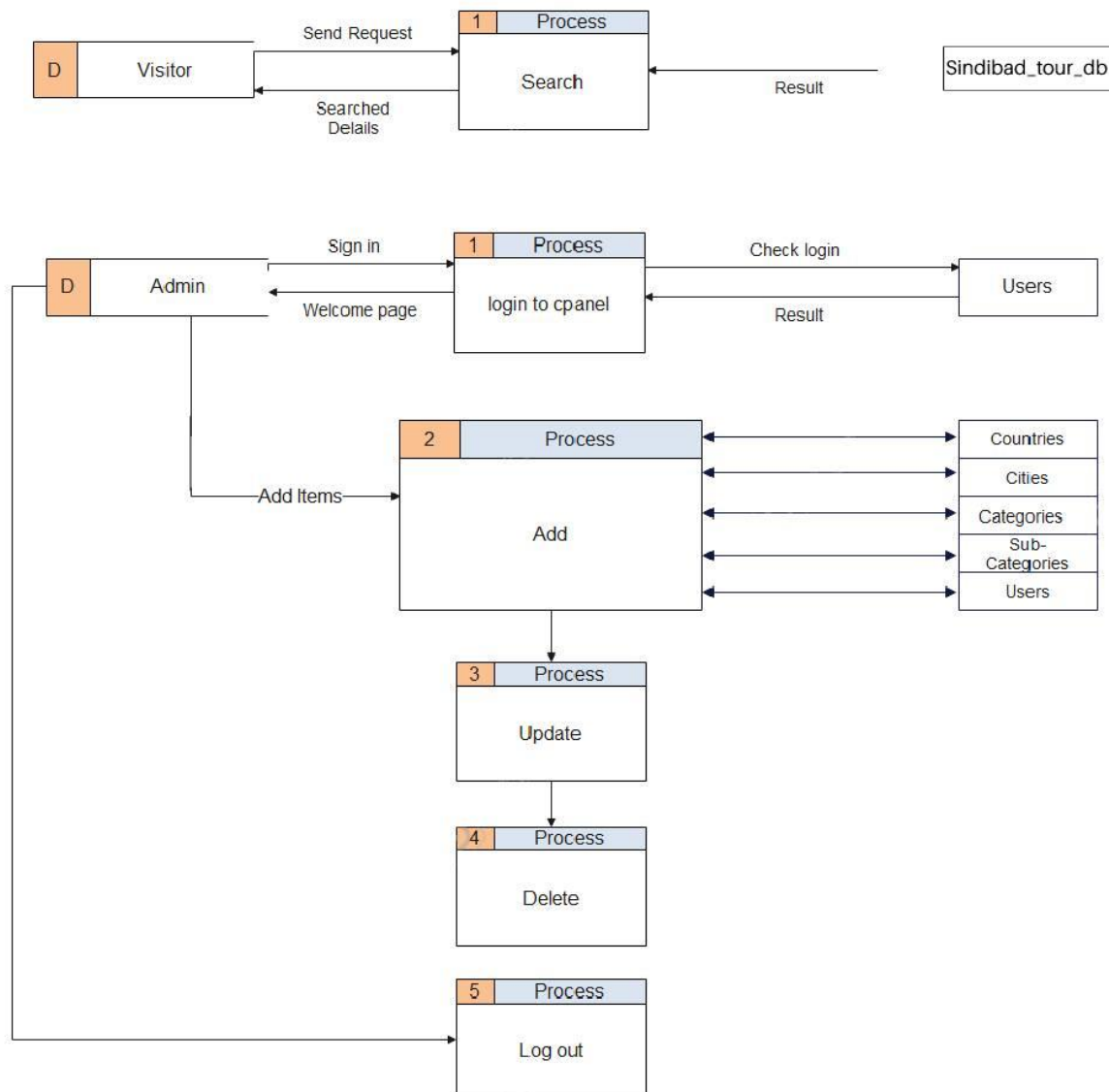
Next use case diagram shows the operations in the whole system, for example the admin can add, edit and delete (CRUD operations) in all tables of database, in addition he can create and manage users of the system, the responsibility of the admin is to make the website sustainable for the tourists or visitors and insert and maintain information in the system. Admin can login to the system with username and password only, no one can enter the system without authentication. In other side, the visitor can enter the website and search for the hotels, restaurants, entertainments and tour guide companies by search engine provided in the first page and get the results. He can browse all services provided by searching by city or country, when he searches by country he will get all services in the cities of that country, if he searches by specific city, he will get services for that city only.



Data flow diagram (DFD):

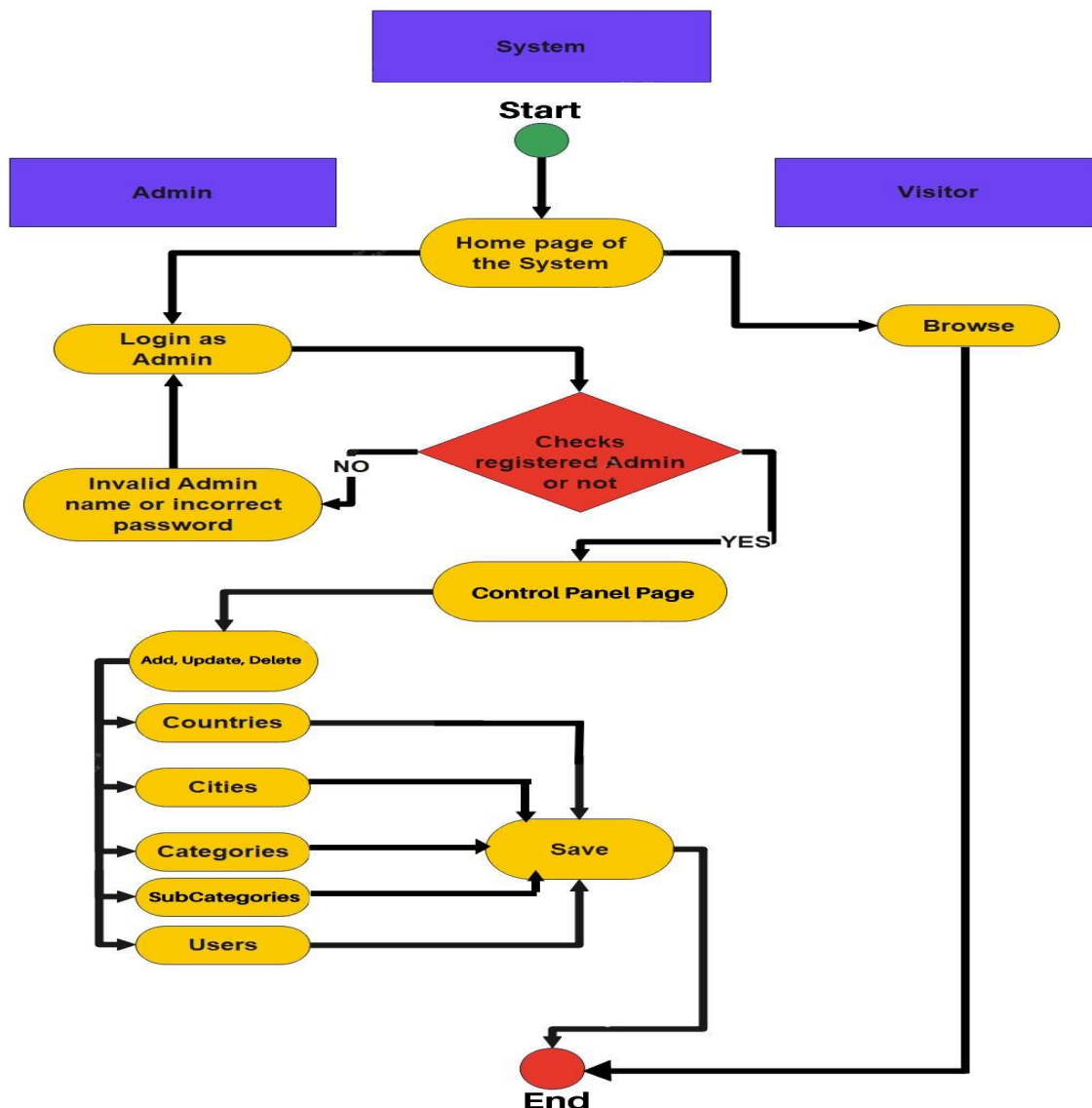
In the next data flow diagram, we can see how the processes done in each part of the system, the DFD diagram consists of two sections, one for visitor, another for admin, in visitor section we can see how data flow from him by sending request to search for hotels, restaurants, entertainments and tour guide companies, then the process treat his request by checking `sindibad_tour_db` warehouse and compare requested info to what is available in the tables, then return search details result to the visitor. The next section is for Admin, Admin send request to warehouse by sing into the system, the process number (1) log in to cPanel treat the admin request and check username and password in users'

warehouse, after check login complete the result return then the process show welcome page for him. After Admin logs in successfully, he can add new items in countries, cities, categories, subcategories and user's warehouses, in addition to update and delete processes can be done by admin for these new entries. In process number (5) the admin can log out from the system by kill the current sessions.



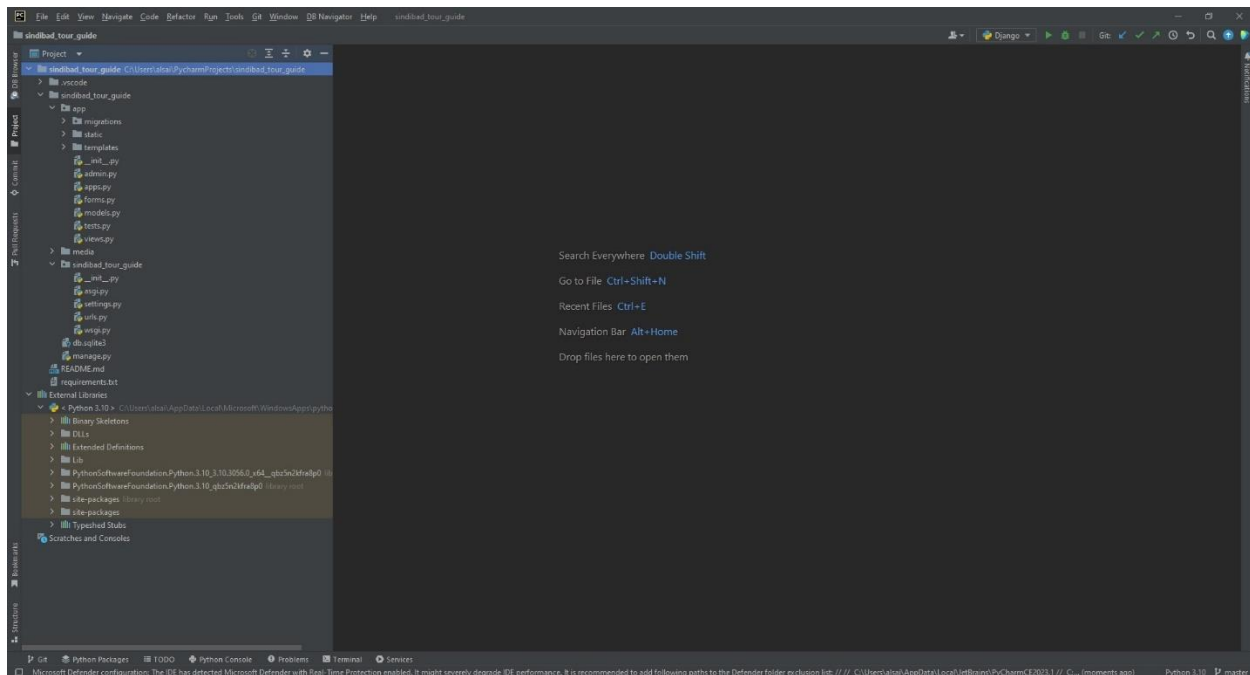
Activity diagram:

The next activity diagram shows the workflow of the system in admin side and visitor side, the visitor just visit homepage of sindibad tour guide website and do search and browse for services then the process end, the admin visit homepage and login to cPanel, the system check registered admin, if the no return to login page again, if yes complete to cPanel page, admin can do add, edit, update processes for countries, cities, categories, subcategories and users, then save all data and end the processes.



Implementation:

The structure of the implementation phase based in MVC, Django framework provides MVC environment that can arrange the coding files with many features, next figure screenshot shows how the structure of Django framework arrange the sindibad tour guide project:



To start a project with Django, first we need to install Django library by next command:

```
PS C:\Users\alsai\PycharmProjects\sindibad_tour_guide\sindibad_tour_guide> pip install django
```

There are many features Django framework provides in his structure related to security, sqlite3 database & SQL injection, URL routing, errors handling, file managing and forms handling. Inside the main directory the application directory contains migrations files and static files in addition to templates (interface) files. The main page of visitors designed by HTML5 and CSS3 designed as the prototype, next code shows the structure of Home.html page of visitors:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Sindibad tour guide search engine</title>
    {% load static %}
    <meta charset="UTF-8" name="viewport" content="width=device-width,
initial-scale=1"/>
    <link rel="shortcut icon" href="{% static 'app/images/sindibad.jpg'
%}" />
    <link href="{% static 'app/css/mystyle.css' %}" rel="stylesheet">
    <script src="{% static 'app/js/myfunction.js' %}"></script>

  </head>
  <body>
    <div class="mynavbar">
      <span style="float: left !important; padding: 2px 2px 2px 8px;
margin: 0;">

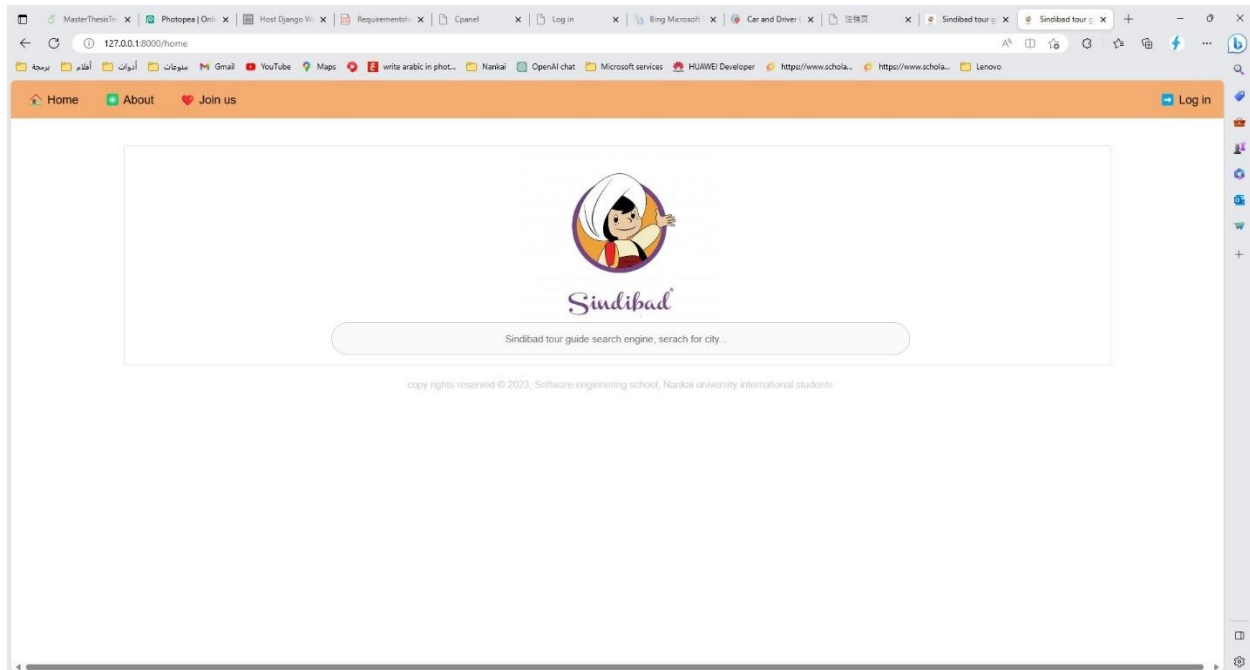
        <a href="/home"><img alt="Home icon" data-bbox="395 350 415 365"/> Home</a>
        <a href="/about"><img alt="About icon" data-bbox="405 368 425 383"/> About</a>
        <a href="#" onclick="joinUs()"><img alt="Join us icon" data-bbox="440 386 460 401"/> Join us</a>
      </span>
      <span style="float: right !important; padding: 2px 2px 2px 8px;
margin: 0;"><a href="/login/index"><img alt="Login icon" data-bbox="475 430 495 445"/> Log in</a></span>

    </div>

    <div class="content rounded border">
      <div class="row">
        <div class="column-12 pad-5 txt-center" id="searchbar">
          {% load static %}
          
          <form method="post" action="/result">
            {% csrf_token %}
            <input type="search" name="city" placeholder="Sindibad tour guide search
engine, serach for city...">
          </form>
        </div>
        {% block content %} {% endblock %}
      </div>
    </div>
    <br>
    <div class="column-12 pad-4 txt-center">
      <span style="color: rgb(207, 207, 207);">copy rights reserved ©
2023, Software enginnering school, Nankai university international
students</span>
    </div>
  </body>
</html>

```

Next figure shows the result of the code in home.html page:



There is models.py file, which is dealing with database tables by converting them to classes. The views.py is the controller of Django framework and contains main methods of the application such as CRUD operation. Next algorithm shows add country method:

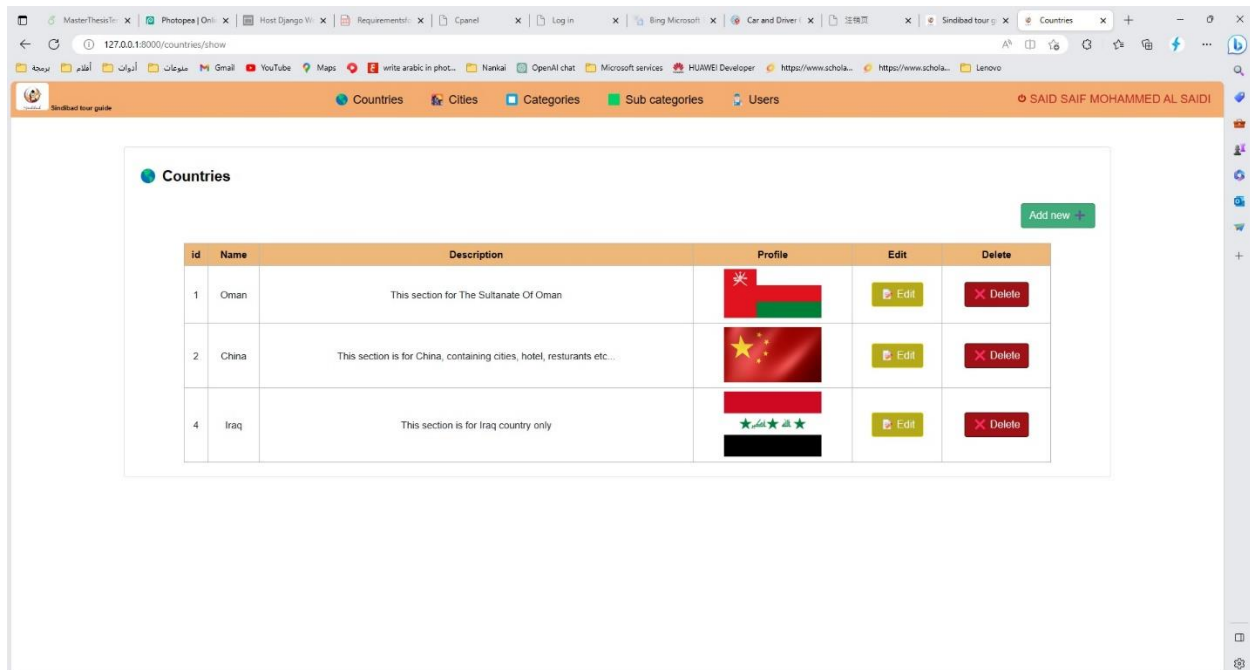
```
def addcountry(request):
    if request.method == "POST":
        form = CountryForm(request.POST, request.FILES)
        if form.is_valid():
            try:
                country = Countries()
                country.active = 1
                country.created_by = request.session['username']
                country.created_date = now
                country.profile = form.cleaned_data['profile']
                country.name = form.cleaned_data['name']
                country.description = form.cleaned_data['description']
                country.save()
                return redirect('/countries/show')
            except:
                pass
        else:
            form = CountryForm()
            return render(request, 'countries/add.html', {'form': form})
```

In `addcountry(request)` method, first thing is checking if user post something, if it's true save everything posted in form `CountryForm`, the second condition is check if the fields in the form are valid by use `is_valid()` built in function in python, then use try and catch to handle exceptions, then we created object of `Countries()` class, insert some fields manually such as `active` to determine is this row deleted or not, and save `created_by` and `created_date` to ensure who save the data and when, and paste every posted fields in their right positions, in the end we use `save()` function to save the data in the table, then redirect user to show page. This method concept implemented in all add methods for cities, categories, subcategories and users models.

Next figure show html page of add country in the system:

To retrieve and show data for of countries model, the next function show how the application retrieve data:

```
def showcountries(request):
    count = Countries.objects.filter(active=1)
    return render(request, 'countries/show.html', {'count':count})
```



So, the count contains all data in Countries, with filtering every active row, then render show.html page in countries directory, then later inside the view (show.html) restructure every row of table inside html table rows, like the next code:

```
<table class="table tb-bordered">
  <thead class="backg-orange">
    <th>id</th>
    <th>Name</th>
    <th>Description</th>
    <th>Profile</th>
    <th>Edit</th>
    <th>Delete</th>
  </thead>
  <tbody>
    {% for country in count %}
    <tr>
      <td>{{country.id}}</td>
      <td>{{country.name}}</td>
      <td>{{country.description}}</td>
      <td>{% if country.profile %}  {% endif %}</td>
      <td><a href="/countries/edit/{{country.id}}" class="butn btn-
yellow"> Edit</a></td>
      <td><a href="/countries/deletecountry/{{country.id}}" class="butn
btn-red"> Delete </a></td>
    </tr>
```

```

        {% endfor %}
    </tbody>
</table>

```

To edit and update country we must get information of specific row in the countries table, then change and update information, next code shows how to get the edit page to implement updating method:

```

def editcountry(request, id):
    count = Countries.objects.get(id = id)
    return render(request, 'countries/edit.html', {'count':count})

```

In this code we notice that we return specified row by his id, count contains data of specified row that the user clicked, then render edit.html in countries directory to edit information.

The screenshot shows a web browser window with multiple tabs. The active tab is 'Edit country'. The URL bar shows '127.0.0.1:8000/countries/edit/2'. The page has an orange header with navigation links: 'Countries', 'Cities', 'Categories', 'Sub categories', and 'Users'. The user's name 'SAID SAIF MOHAMMED AL SAIDI' is displayed on the right. The main content area is titled 'Countries' and contains a form labeled 'Edit details:'. The form has the following fields:

- ID:** A text input field containing the value '2'.
- Name:** A text input field containing the value 'China'.
- Description:** A large text area containing the text 'This section is for China,'.
- Icon:** A file upload field with a 'Choose File' button and the text 'No file chosen'.

At the bottom of the form is a yellow 'Update' button.

After user click update in edit page, the form send all data to updatecountry() method to change and update information, next code show the process of update:


```
def updatecountry(request, id):
    count = Countries.objects.get(id = id)

    form = CountryForm(request.POST, request.FILES, instance=count)
    if form.is_valid():
        count.created_by = count.created_by
        count.created_date = count.created_date
        count.active = 1
        count.updated_by = request.session['username']
        count.updated_date = now
        count.profile = form.cleaned_data['profile']
        count.name = form.cleaned_data['name']
        count.description = form.cleaned_data['description']
        count.save()
        return redirect('/countries/show')
    return render(request, 'countries/edit.html', {'count':count})
```

The first thing is to bring the old information of the row, then paste the request.POST in the CountryForm, check if all entries valid in the form by using is_valid() built in method, then change old information to the new information and save the changes. After that, redirect the user to countries/show page again. If the condition is not working, the system will return the user to edit page again. This is implemented in all models with the same algorithm.

To delete country, the deleting concept means updating active field by set zero or one, if its value zero that means the row is deleted or hidden, if its value one it means the row is active and visible to the user, this strategy is good when we want to keep history of database, and protect data and avoid associated problems (FK indexes) when implemented deleting concept literally. Next code shows the main method to delete row of countries table:

```
def deletecountry(request, id):
    count = Countries.objects.get(id = id)
    count.created_by = count.created_by
    count.created_date = count.created_date
    count.updated_by = request.session['username']
    count.updated_date = now
    count.active = 0
    count.save()
    return redirect('/countries/show')
```

The code is similar to the previous code (updatecountry() method), with just one change which is set the value of active field from one to zero, then save all information and redirects user to show.html page, other models use same concept to for deleting rows.

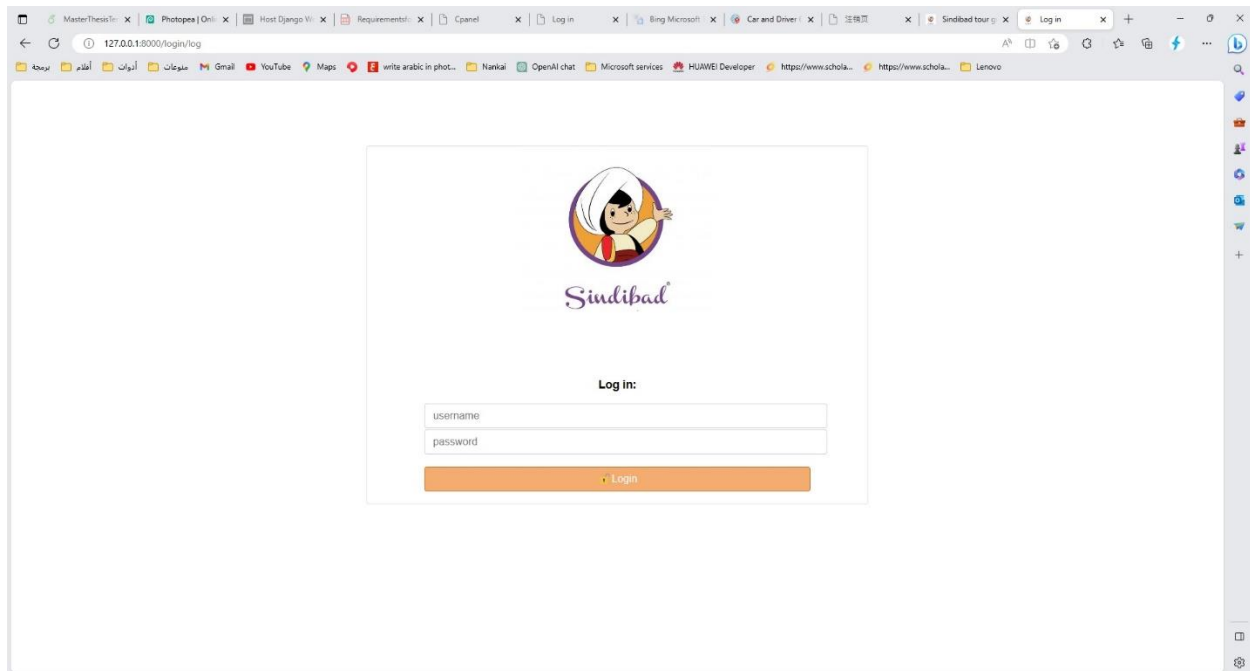
There is also important methods like singin(request) method to check the username and password are correct and not, next algorithm shows the code of the method singin():

```
def singin(request):
    if request.method == "POST":
        form = UserForm(request.POST)
        if form.is_valid():
            try:
                check =
Users.objects.filter(username=form.cleaned_data['username']).filter( password
= form.cleaned_data['password']).filter(active = 1)
                if not check:
                    request.session['username'] = 'nothing'
                    messages.info(request, 'Your username or password are
wrong!')
                    return render(request,'login/log.html')
                else:
                    request.session['username'] =
form.cleaned_data['username']
                    for i in check:
                        request.session['name'] = i.name
                        messages.info(request, 'Login successfully !')
                        return render(request,'index.html')
            except:
                pass
        else:
            form = UserForm()
            return render(request,'/login/log.html',{'form':form})
```

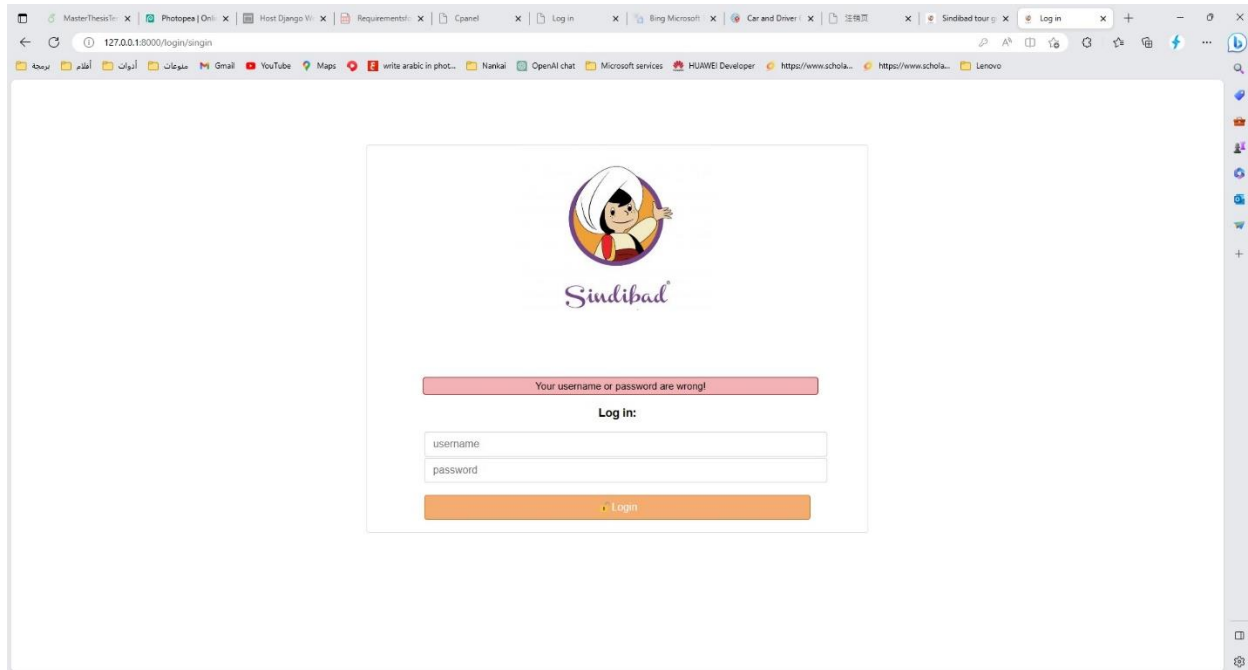
The first condition is to check if the user post something, if its true it will continue to save all posted information inside UserForm, then check if the posted information in the form is valid, after that the system will check if the username and password are correct by checking that inside the query, if there is no result, the user will redirected to login/log.html page again, if its ok, the user will redirected to index.html page of cPanel, before redirecting user there will initializing of

session['username'] and session['name'] to register everything done under his control.

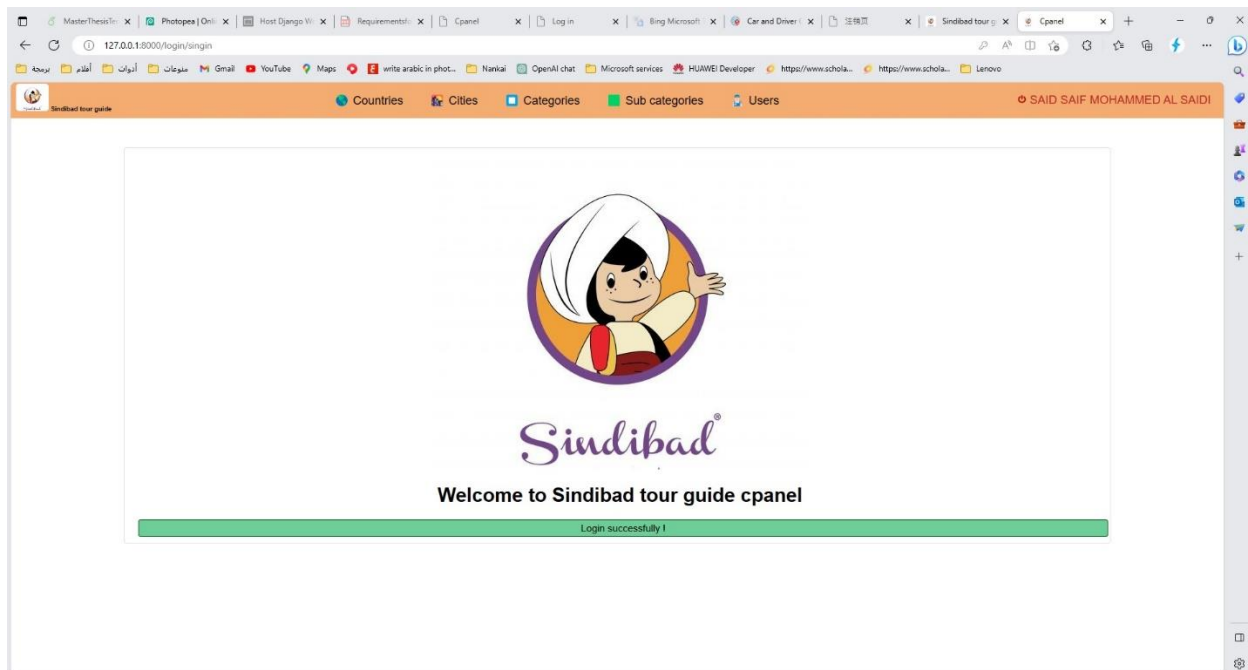
The next figure shows the login form page, its separated html page to let user sign in in the system:



If the user tries to insert wrong username or password the page will show error result, the next figure show test of wrong username or password:



If everything is ok, the system will redirect the user to home page of cPanel:



When visitor tries to search for services, the system call method called `result()`, this method will filter the keywords that visitor entered (country name or city name), the next code explains the main algorithm of Sindibad search engine:

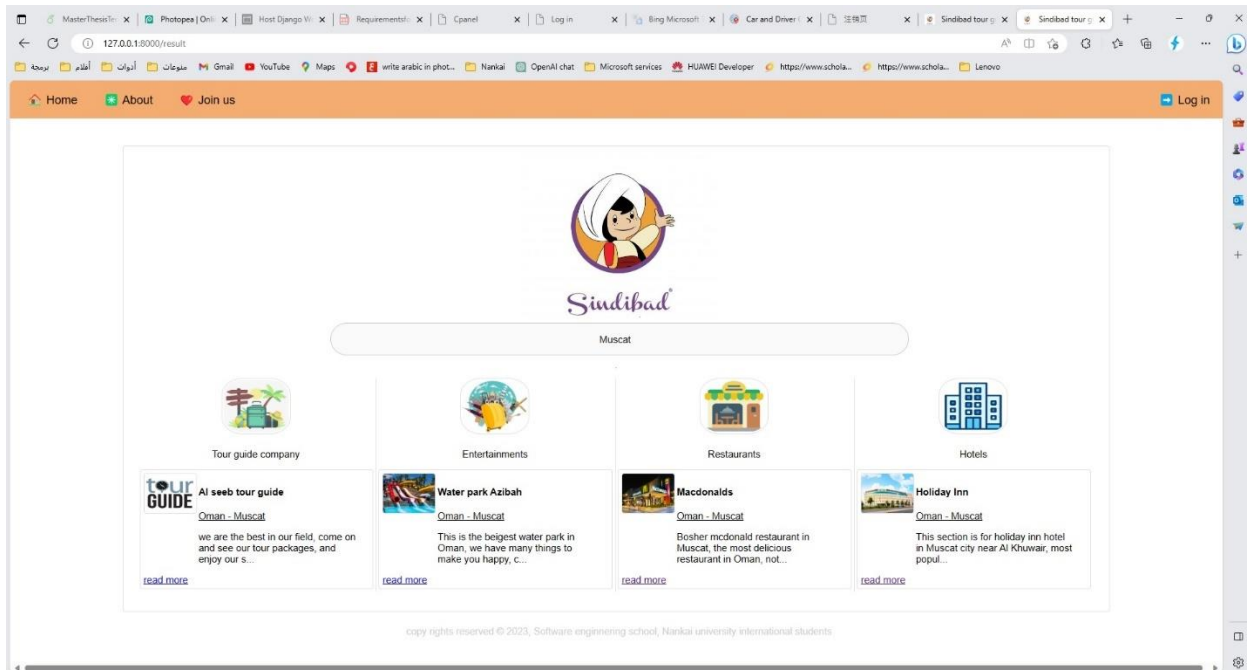
```

def result(request):
    if request.method == "POST":
        # get the keyword of search
        city = request.POST['city']
        # check first in citis if the keyword is city
        c = Cities.objects.filter(name__contains = city)
        if c.count() >0:
            # search in every sub_category for that city
            for i in c:
                cc = i.id
                hotels = SubCategories.objects.filter(city_id = cc,cat=9,active
=1)
                restaurants = SubCategories.objects.filter(city_id =
cc,cat=10,active =1)
                entertainments = SubCategories.objects.filter(city_id =
cc,cat=12,active =1)
                tour_guide = SubCategories.objects.filter(city_id =
cc,cat=13,active =1)
                return
        render(request,'result.html',{'hotels':hotels,'restaurants':restaurants,'ente
rtainments':entertainments,'tour_guide':tour_guide})
    else:
        # check if the posted keyword is country
        country = request.POST['city']
        # search in countries
        co = Countries.objects.filter(name__contains = country)
        hotels = ""
        restaurants = ""
        entertainments = ""
        tour_guide = ""
        if co.count() >0:
            for ii in co:
                # search for cities in that country
                id_country = Cities.objects.filter(country = ii.id)
                if id_country.count()>0:
                    for ci in id_country:
                        # here we save all results in same hotels,
                        restaurants, entertainments, tour_guide by using chain library to concatenate
                        QuerySet
                        hotels =
chain(hotels,SubCategories.objects.filter(city_id = ci.id,cat=9,active =1))
                        restaurants =
chain(restaurants,SubCategories.objects.filter(city_id = ci.id,cat=10,active
=1))
                        entertainments =
chain(entertainments,SubCategories.objects.filter(city_id =
ci.id,cat=12,active =1))
                        tour_guide =
chain(tour_guide,SubCategories.objects.filter(city_id = ci.id,cat=13,active
=1))
                    return
        render(request,'result.html',{'hotels':hotels,'restaurants':restaurants,'ente
rtainments':entertainments,'tour_guide':tour_guide})
        return render(request,'home.html')
    else:
        return render(request,'home.html')

```

First, the method will try to search for city in cities model class, if everything ok, the system will move to next condition to get every service related to that city (hotels, restaurants, entertainments, tour guide companies) by searching inside subcategories model and filtering the information by category id and city id. If the condition not true, the method will try another way by searching by countries to retrieve all information related to their cities, first the query to search the country, then if the condition is true, the method will search for cities and make nested loops the get all services inside the cities, and thus the search conditions is achieved and bring the result for the visitor.

When user search for something, the results will be structured like the next page:



For example, the visitor search for Muscat city capital of the Sultanate of Oman, if there are information stored about this city the system will bring them like in the figure.

To control URLs routing, there is file called `url.py` inside Django framework to control redirections and routing inside `views.py`, the next code explains how this is done:

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.homepage, name='index'),
    path('home', views.homepage, name='index'),
    path('login/log', views.loginpage, name='log'),
    path('index', views.indexpage, name='index'),
    path('result', views.result, name='result'),
    path('about', views.about, name='about'),
    path('page/<int:id>', views.page, name='page'),
    path('countries/show', views.showcountries, name='show'),
    path('cities/show', views.showcities, name='show'),
    path('users/show', views.showusers, name='show'),
    path('categories/show', views.showcategories, name='show'),
    path('sub_cat/show', views.showsubcat, name='show'),
    path('login/index', views.logout, name='log'),
    path('login/singin', views.singin, name='singin'),
    path('login/log', views.singin, name='singin'),
    path('countries/add', views.addcountry, name='addcountry'),
    path('users/add', views.adduser, name='adduser'),
    path('cities/add', views.addcity, name='addcity'),
    path('categories/add', views.addcategory, name='addcategory'),
    path('sub_cat/add', views.addsubcat, name='addsubcat'),
    path('countries/edit/<int:id>', views.editcountry, name='editcountry'),
    path('users/edit/<int:id>', views.edituser, name='edituser'),
    path('cities/edit/<int:id>', views.editcity, name='editcity'),
    path('categories/edit/<int:id>', views.editcategory,
name='editcategory'),
    path('sub_cat/edit/<int:id>', views.editsubcat, name='editsubcat'),
    path('countries/updatecountry/<int:id>', views.updatecountry,
name='updatecountry'),
    path('users/updateuser/<int:id>', views.updateuser, name='updateuser'),
    path('cities/updatecity/<int:id>', views.updatecity, name='updatecity'),
    path('categories/updatecategory/<int:id>', views.updatecategory,
name='updatecategory'),
    path('sub_cat/updatesubcat/<int:id>', views.updatesubcat,
name='updatesubcat'),
    path('countries/deletecountry/<int:id>', views.deletecountry,
name='deletecountry'),
    path('cities/deletecity/<int:id>', views.deletecity, name='deletecity'),
    path('categories/deletecategory/<int:id>', views.deletecategory,
name='deletecategory'),
```

```

    path('sub_cat/deletesubcat/<int:id>', views.deletesubcat,
name='deletesubcat'),
    path('users/deleteuser/<int:id>', views.deleteuser, name='deleteuser'),
] + static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)

```

In addition, there is settings.py file, this is very important file to control many things inside project, such as directories, uploaded files, database connections and database provider, the next code shows how to set connection to MySQL workbench database inside settings.py:

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'sindibad_tour_db',
        'USER': 'myuser',
        'PASSWORD': 'mypwd',
        'HOST': '127.0.0.1',
        'PORT': '3306'
    }
}

```

Testing phase:

Django provides a testing framework that allows to write tests for the application. Tests in Django are based on the unittest module. We implemented test locally, by using test.py file and initialize class called FooTest(TestCase), the run it by open a terminal, and navigate to the project's root directory, and execute the following command `python manage.py test`, the next figure show the result of local testing:

```

Terminal: Local
Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\alsai\PycharmProjects\sindibad_tour_guide> cd sindibad_tour_guide
PS C:\Users\alsai\PycharmProjects\sindibad_tour_guide\sindibad_tour_guide> python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Win
.
-----
Ran 1 test in 0.002s

OK
Destroying test database for alias 'default'...
PS C:\Users\alsai\PycharmProjects\sindibad_tour_guide\sindibad_tour_guide>

```


The result shows everything is ok with python codes, there are no errors generated by running the command 'python manage.py test'. The next code shows class:

FooTest:

```
from django.test import TestCase

# Create your tests here.
class FooTest(TestCase):
    def setUp(self):
        pass

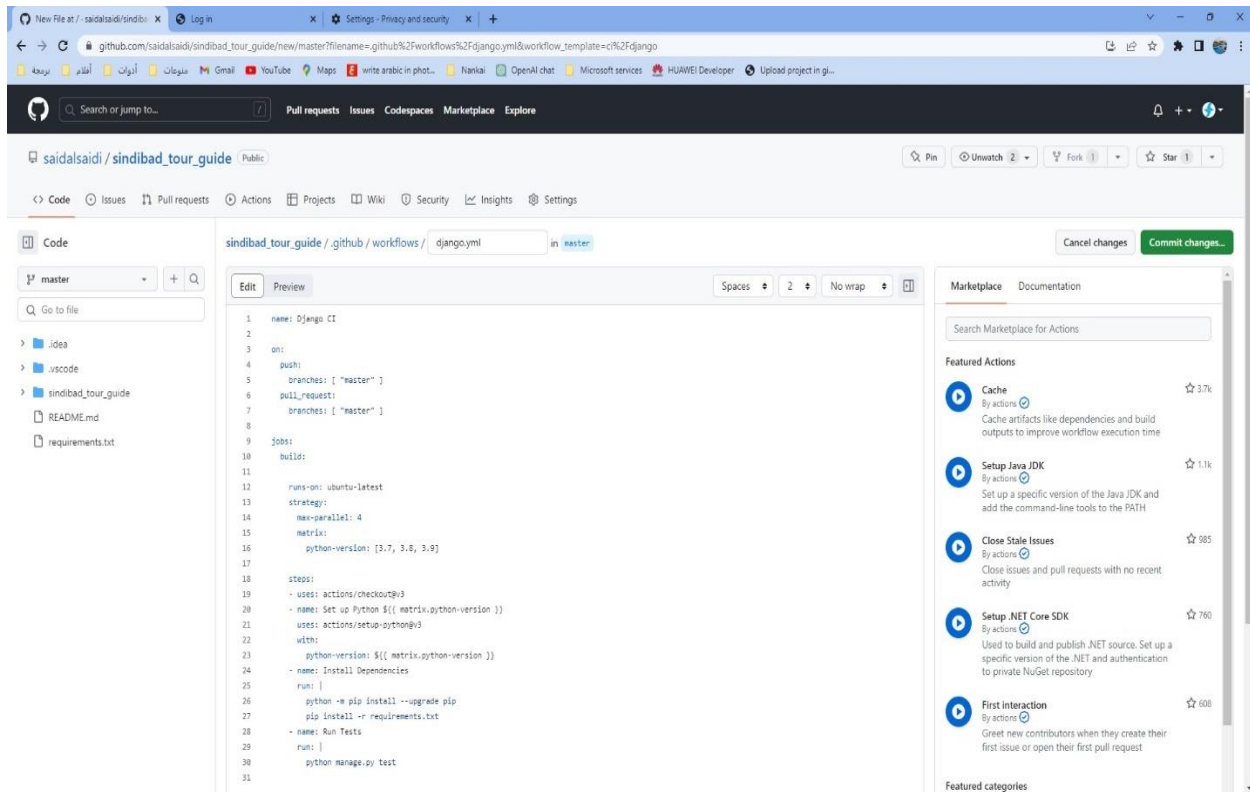
    def tearDown(self):
        pass

    def this_wont_run(self):
        print('Fail')

    def test_this_will(self):
        print('Win')
```

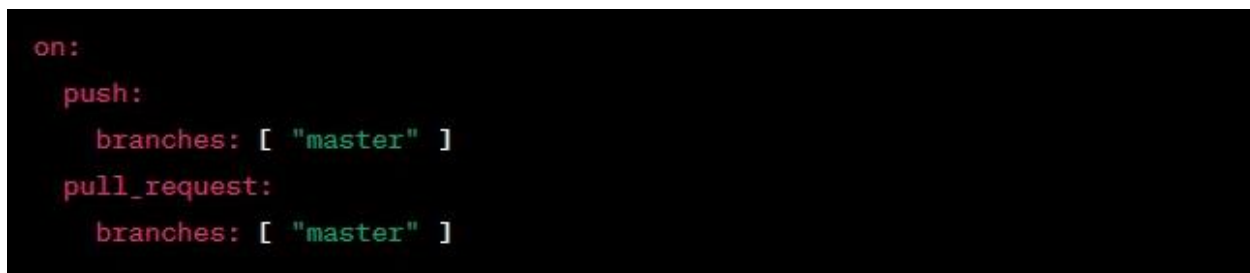
Also, the project tested in GitHub by using Django CI, this tool allowed us to define workflows that run our tests automatically whenever changes are pushed to the (sindibad_tour_guide) repository. This tool provides visibility into test results, enables collaboration, and helps catch errors early in the development process, The next code show structure of CI tool in GitHub for our project.

CI tool:



This specific workflow is named "Django CI" and is triggered when there is a push to the "master" branch or a pull request targeting the "master" branch.

Let's break down the code and understand its different sections:



This section defines the events that trigger the workflow. In this case, it triggers when there is a push to the "master" branch or a pull request targeting the "master" branch.

```
jobs:
  build:
    runs-on: ubuntu-latest
    strategy:
      max-parallel: 4
      matrix:
        python-version: [3.7, 3.8, 3.9]
```

This section defines the job named "build" that will run in this workflow. It specifies that the job will run on the latest version of Ubuntu. The "strategy" section allows you to run parallel jobs with different configurations. In this case, it sets up a matrix to run the job with different Python versions (3.7, 3.8, and 3.9).

```
steps:
  - uses: actions/checkout@v3
  - name: Set up Python ${ matrix.python-version }
    uses: actions/setup-python@v3
    with:
      python-version: ${ matrix.python-version }
  - name: Install Dependencies
    run: |
      python -m pip install --upgrade pip
      pip install -r requirements.txt
  - name: Run Tests
    run: |
      python manage.py test
```

This section lists the steps to be executed within the "build" job:

- The first step (actions/checkout@v3) checks out the code from the repository.
- The second step sets up the specified Python version using actions/setup-python@v3.

- The third step installs the project dependencies by upgrading pip and installing the packages listed in the requirements.txt file.
- The fourth step runs the Django tests using python manage.py test.

Deployment:

The application is designed to web environment, to make it available for all users, the most important things that must be available in the hosting environment is support for the latest versions of Python that are compatible with the project (python 3.10) and support MySQL databases, pythonanywhere.com is the deployment environment to host the application, PythonAnywhere is a cloud-based Python development and hosting platform that allows to run and deploy Python web applications, scripts, and scheduled tasks in a server environment without the need for local installation or server management. It provides an integrated development environment (IDE), a range of Python libraries and frameworks, and the ability to host and serve web applications.

Key features and services provided by PythonAnywhere include:

- Web Hosting: PythonAnywhere allows you to deploy and host web applications written in Python. It provides a web server environment where you can upload your code, configure the necessary settings, and access your application via a custom domain or PythonAnywhere subdomain.
- Web App Frameworks: PythonAnywhere supports popular Python web frameworks such as Django, Flask, and Pyramid. You can easily deploy and run applications built using these frameworks.
- Scheduled Tasks: You can schedule Python scripts or tasks to run at specified intervals using PythonAnywhere's scheduler. This is useful for automating repetitive tasks or performing periodic data updates.

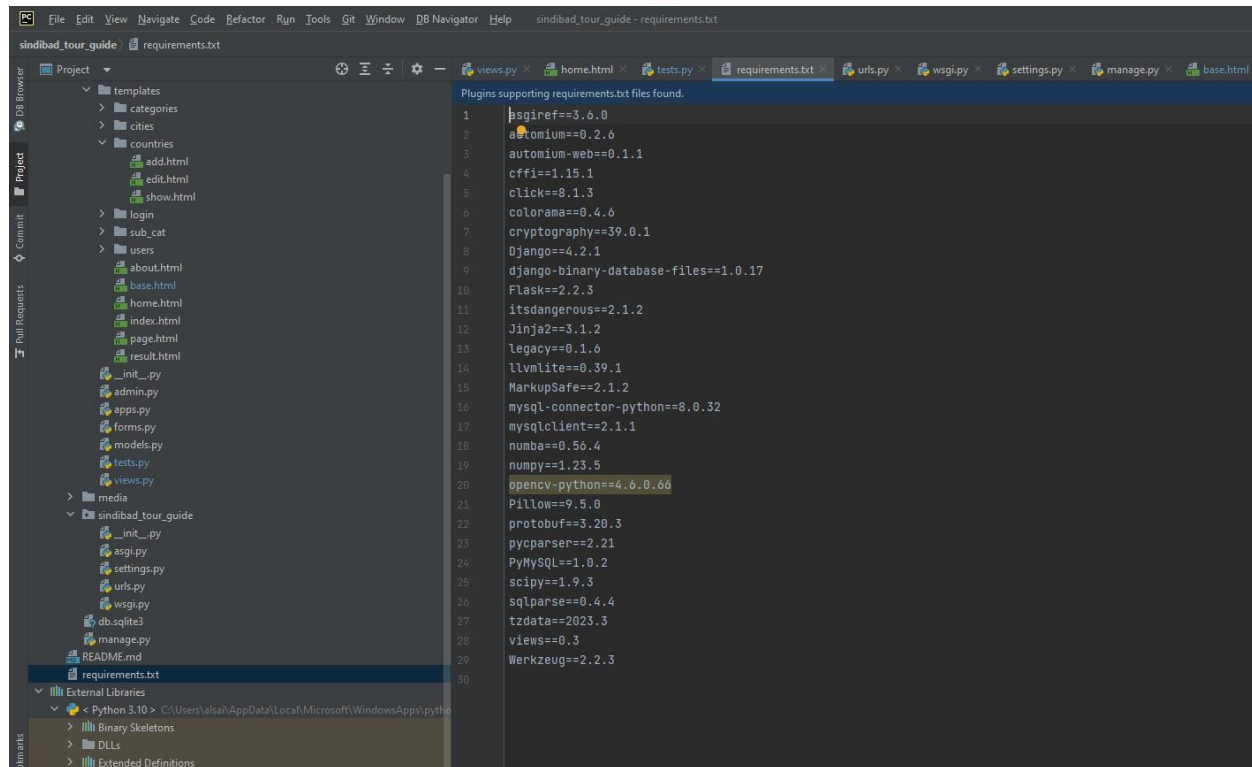
- **Interactive Console:** PythonAnywhere provides an interactive Python console, also known as a "Bash Console" where you can execute Python code and interact with your server environment.
- **File Management:** The platform allows you to upload, edit, and manage your project files through an integrated file explorer. You can also access your files via FTP/SFTP.
- **Database Integration:** PythonAnywhere supports various database systems, including MySQL, PostgreSQL, and SQLite. You can configure and manage databases for your applications directly from the platform.
- **Collaboration and Sharing:** PythonAnywhere facilitates collaboration among team members by allowing multiple users to work on the same project simultaneously. You can also share your web applications or code snippets with others.
- **Monitoring and Logs:** PythonAnywhere provides access to application logs and server monitoring tools to help you track and troubleshoot issues.

PythonAnywhere offers free and paid subscription plans, allowing you to choose the level of resources and features that suit your needs. It provides a user-friendly interface and simplifies the process of deploying and managing Python applications in a server environment.

After creating account in pythonanywhere.com and before we published the website to pythonanywhere.com, we need to prepare the system requirements of our project by running next line in terminal:

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

This command will generate requirements.txt file and provide all the libraries needed to let the application work correctly in deployment environment, the next figure show the result of the command for our project:



Here's an overview of the steps involved:

- Upload your code to Hosting Cloud Server
- Set up a virtualenv and install Django and any other requirements.
- Set up web app using the manual config option.
- Add any other setup (static and media files, env variables).

Uploading Django Code to Hosting Server:



As our code is ready on GitHub, we will clone it using Bash Console. Create a Bash Console by clicking on the Console Tab. We will see the terminal interface where we need to type git clone command:

```
git clone https://github.com/saidalsaidi/sindibad_tour_guide.git
```

Create VirtualEnv and Install Django and Dependencies:

In Bash console, we will create a virtualenv, named after your project name, and choose the version of Python you want to use like we are using now 3.10:

```
mkvirtualenv --python=/usr/bin/python3.10 mysite-virtualenv
```

There will be terminal like this – (mysite-virtualenv) \$ which means virtualenv is activated. If the terminal not shown, then we will type bellow command which will activate it:

```
workon mysite-virtualenv
```

Once the Virtual Env is activated. We will install Django inside it. There are two ways to do it. If we generate requirements.txt file, then we can use following:

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

Or simply install Django and Required Libraries:

```
pip install django
```

Setting up Django Web app and WSGI file:

At this point, we need to be armed with 3 pieces of information:

- 1- The path to Django project's top folder — the folder that contains “manage.py”, eg – /home/alsaidisoft/sindibad_tour_guide
- 2- The name of the project (that's the name of the folder that contains settings.py), eg app in our project.
- 3- The name of our virtualenv, eg app-virtualenv.

Create a Web Application with Manual Config:

We will go to Web Tab and Click on it. We will see create a web application button, we will click on it and follow instructions:

Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

- » Django
- » web2py
- » Flask
- » Bottle
- » **Manual configuration (including virtualenvs)** **Click Here**

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

Once our web app is created, we scroll down, and we need to add some paths like shown in bellow image:

Code:

What your site is running.

Source code: /home/alsaidisoft/sindibad_tour_guide/sindibad_tour_guide/ [Go to directory](#) **1**

Working directory: </home/alsaidisoft/> [Go to directory](#) **2**

WSGI configuration file: /var/www/alsaidisoft_pythonanywhere_com_wsgi.py

Python version: 3.10 [🔧](#)

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

</home/alsaidisoft/.virtualenvs/app-virtualenv> **3**

[🔧 Start a console in this virtualenv](#)

We should make sure folders' names accordingly to our project names:

- 1- Code – /home/alsaidisoft/sindibad_tour_guide/.
- 2- Virtualenv – /home/alsaidisoft/.virtualenvs./mysite-virtualenv.
- 3- virtual env name.

Edit WSGI File to Point our Django Project to Server:

Code:

What your site is running.

Source code: /home/alsaidisoft/sindibad_tour_guide/sindibad_tour_guide/ [Go to directory](#)

Working directory: </home/alsaidisoft/> [Go to directory](#)

WSGI configuration file: /var/www/alsaidisoft_pythonanywhere_com_wsgi.py

Python version: 3.10 [🔧](#)

We click on the WSGI configuration file link. It will take us to our Hosted Django Project WSGI File. We need to add the following code. Just remove everything inside that file and paste the bellow code in it:

```

import os
import sys

# assuming our Django settings file is at
# '/home/alsaidisoft/sindibad_tour_guide/sindibad_tour_guide/settings.py'
path = '/home/alsaidisoft/sindibad_tour_guide'
if path not in sys.path:
    sys.path.insert(0, path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'sindibad_tour_guide.settings'

from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()

```

Create MySQL database for our website:

The next step is to create database, we already create MySQL database for our website called sindbad_tour_db from database tab:

MySQL settings

Connecting:

Use these settings in your web applications.

Database host address:	alsaidisoft.mysql.pythonanywhere-services.com
Username:	alsaidisoft

Your databases:

Click a database's name to start a MySQL console logged in to it.

Name

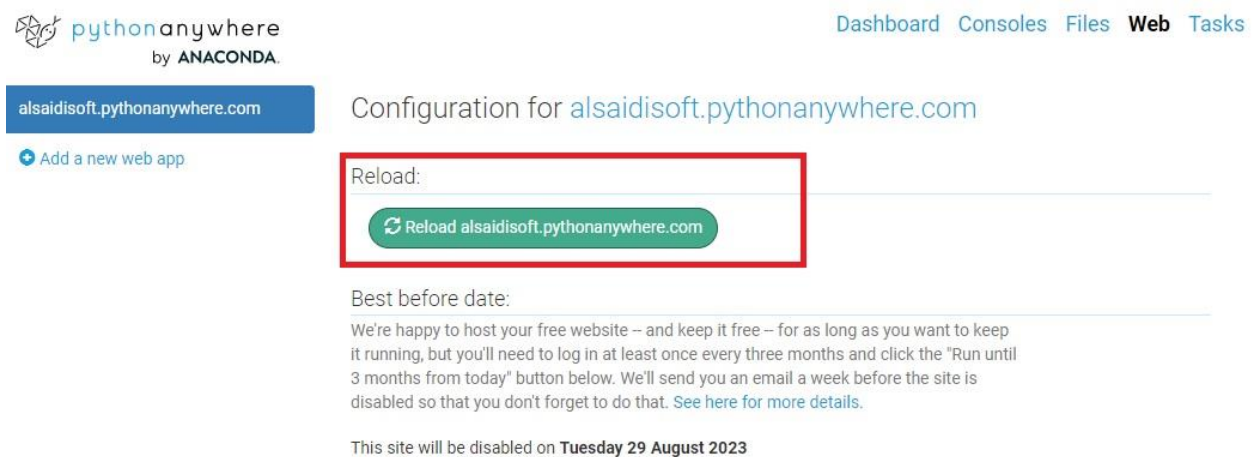
alsaidisoft\$default

alsaidisoft\$sindibad_tour_db

We need to change database settings name and hosting from setting.py and set the new hosting name, username and password, and we need also to disallow host error by next command in settings.py:

```
ALLOWED_HOSTS =['*']
```

That's it. Now the file step is to go to Web Tab and reload the web app and click on the link of our web app:



Conclusion:

Sindibad tour guide search engine is website to make it easy for tourist and other users to search for anything related to their interest in tourism, the app will help to get what the customers need to get best tour in the cities, instead of using many apps and consume time of registering and accessing all these platforms to check hotels, restaurants, entertainments and tour guide companies.

This software engineering documentation has provided a comprehensive overview of the project, its requirements, design, implementation, testing, and maintenance. Throughout the documentation, we have highlighted key

considerations, challenges, and solutions encountered during the software development lifecycle.

The project successfully addressed the identified requirements and delivered a robust and functional software solution. The collaborative efforts of the development team, stakeholders, and end-users have been crucial in achieving the desired outcomes.

Moreover, the documentation has highlighted the significance of documentation itself. Clear and comprehensive documentation serves as a valuable resource for developers and maintainers, enabling them to understand the system's architecture, functionalities, and maintenance processes.

While the project has achieved its goals, it is important to acknowledge that software development is an ongoing process. As technology evolves and new requirements emerge, the software will need to be regularly updated and enhanced to meet changing needs. This requires a commitment to ongoing maintenance, bug fixes, and potential feature additions.

In conclusion, this software engineering documentation serves as a comprehensive record of the project's lifecycle, capturing the decisions, challenges, and solutions encountered during the development process. It provides a solid foundation for future development, maintenance, and further enhancements to the software. The successful completion of this project is a testament to the collaborative efforts and adherence to software engineering best practices by the development team.