**A Project Report on**

**EV CHARGING STATION FINDER**

**AND SLOT BOOKING SYSTEM**

Submitted in partial fulfilment of requirement for the award of

**DEGREE NAME**

**Of**

**UNIVERSITY**

**By**

**ROLL NO**

INSTITUTE

ADDRESS

20XX-20XX

**UNIVERSITY**

ADDRESS

**CERTIFICATE**

This is to certified that this project entitled “EV CHARGING STATION FINDER AND SLOT BOOKING SYSTEM” is a bonafide record of the project work done by XXXX with Reg No XXXX, Xrd semester in partial fulfilment of the award of Degree from University during the year 20XX-20XX.

Staff in charge: Assistant Director

Place: External Examiner

Date: 1.

2.

**DECLARATION**

I XXXX, of Xrd Semester XX student of Institute, do hereby declare that the project report on “EV CHARGING STATION FINDER AND SLOT BOOKING SYSTEM” is the original work carried out by me, under the supervision and guidance of Mr xxxx. This project report submitted on the partial fulfilment of the during the period of study 20xx- 20xx.

Name: xxxxxxxxxxx Signature:

# ACKNOWLEDGEMENT

# First of all, I express my sincere thanks to Mrs. xxxx for giving permission for the fulfilment of this venture. My profound gratitude goes to Internal Guide for her enormous goodwill, valuable guidance and generous help without which a mini project work could not have been possible. I am thank full to all faculty members in the department for their cooperation towards this mini project fulfilment.

# At last, but not the least I would like to express my thanks to the omnipotent God who gave me the strength to complete this project, and also sincere thanks to my friends, who have supported me in the project, and encourage much.

**ABSTRACT**

This project aims to design and develop an electric vehicle (EV) charging station and slot booking system to facilitate the increasing number of EVs on the road. The system will provide users with the ability to find and book available charging slots at the nearest charging station, ensuring a hassle-free charging experience. This EV charging station is currently available in all Kerala district, India. User can select from the list and book accordingly.

The system is designed to help electric vehicle owners locate available charging stations and book charging slots.

The frontend of the system is designed using HTML and CSS to provide an interactive and user-friendly interface for EV owners. The frontend allows users to search for nearby charging stations, view charging station. Users can also book a charging slot by selecting the desired charging station and available slot.

The backend of the system is built using Python programming language and utilizes various libraries and Flask frameworks to interact with the frontend and database. The backend handles user authentication, charging station management, and booking management.

The system's database is built using MYSQL to store information about the charging stations, their locations, and available slots. The database is updated in real-time to ensure accurate information is displayed to the user. The database stores user information, including booking details, charging station information, and other relevant information required to operate the system.

The system provides a comprehensive user management module that enables users to create an account, login, and manage their booking history. The user management module also provides additional features such as password reset, email verification, and profile update.

In conclusion, the electric vehicle charging station finder and slot booking system is a comprehensive and user-friendly application that helps EV owners find and book charging slots easily. The system provides accurate and real-time information about charging stations and their availability, making it convenient for EV owners to charge their vehicles.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SERIAL NO** | **TITLE** | **PAGE NO** |
| 1 | INTRODUCTION | 1 |
| 2 | SYSTEM ANALYSIS | 2 |
| 3 | SYSTEM REQUIREMENTS | 3 |
| 4 | FEASIBILITY STUDY | 4 |
| 5 | TECHNOLOGIES USED BEHIND | 8 |
| 6 | SYSTEM DESIGN | 11 |
| 7 | SYSTEM IMPLEMENTATION AND TESTING | 26 |
| 8 | SYSTEM MAINTANANCE | 33 |
| 9 | SCREEN SHOTS | 34 |
| 10 | SOURCE CODE | 38 |
| 11 | CONCLUSION | 44 |
| 12 | BIBILOGRAPHY | 45 |

**INTRODUCTION**

The advent of electric vehicles (EVs) has brought about a paradigm shift in the automotive industry, as these vehicles offer a more environmentally friendly and sustainable mode of transportation. However, one of the major challenges faced by EV owners is the availability of charging infrastructure. The lack of an efficient and reliable charging network often discourages potential buyers from purchasing electric vehicles. To address this issue, we propose the development of an electric vehicle charging station finder system website that will help EV owners locate charging stations and book them in advance.

The proposed system will be a web-based application that will be accessible to all EV owners. It will provide users with real-time information on the location, availability, and type of charging stations in their vicinity. The system will also allow users to book charging stations in advance, ensuring that they are available when they need them. Users will be able to select their preferred charging station and reserve it for a specific time slot.

The proposed system has several advantages over existing EV charging station finder systems. Firstly, it will provide users with accurate and up-to-date information on the availability and type of charging stations, reducing the chances of EV owners being stranded without a charge. Secondly, the system will allow users to book charging stations in advance, reducing waiting times and ensuring that charging stations are available when needed.

In conclusion, the proposed electric vehicle charging station finder system website is an innovative solution to the challenges faced by EV owners in locating and booking charging stations. The system will provide accurate and up-to-date information on the location, availability, and type of charging stations, while also allowing users to book charging stations in advance. The proposed system has the potential to revolutionize the way EV owners find and use charging stations, making electric vehicles a more practical and convenient mode of transportation.

**SYSTEM ANALYSIS**

System analysis is the process of gathering and interpreting facts, diagnosing problems and facts to improve the system. Analysis is detailed study of various operations performed by a system and their relationships within and outside of the system. This involves gathering information and using structured tools for analysis.

**Existing system**

The existing system for electric vehicle charging station finder involves the use of various mobile applications and websites that provide information about the available charging stations. However, these applications often lack crucial features such as the ability to book a charging slot in advance and view detailed information about the charging station, such as the type of charging port available.

**Proposed system**

The proposed system is designed to be user-friendly and efficient, with a simple and intuitive interface that makes it easy for users to navigate and use. The system includes the following features:

1. **Charging station search:**

EV owners can search for nearby charging stations by selecting a location from a list. The system will display a list of charging stations in the area, along with information such as an address, city, charge type, and available slots.

1. **Charging station details:**

Users can view details of a charging station such as its address, city, charge type, and available slots.

1. **Slot booking:**

Users can book available charging slots at their preferred charging station. The system will display a list of available slots, and users can select the slot that best suits their needs. Users can also cancel their booking if they need to.

1. **User management:**

Users can create an account, log in, and manage their booking history. The user management module also provides additional features such as password reset, and email verification.

1. **Admin panel:**

Charging station operators can log in to an admin panel to manage their charging stations and slots. The admin panel allows operators to add or remove charging stations, update charging station details, and view booking history.

# SYSTEM REQUIREMENTS

The minimal requirement for the working of this system is given bellow:

**Hardware Requirements**

* Processor 64 bit
* RAM Min 3 GB
* Hard disk 100 GB

**Software Requirements**

One of the most difficult task is selecting software for the system, once the system requirements is found out when we have to determine whether a particular software package fits for those system requirements.

* Operating system windows 7 or above, android
* Technology used Python
* IDE PyCharm ,Eclipse/ AndroidStudio
* Frame work Flask
* Database MYSQL

**FEASIBILITY STUDY**

A feasibility analysis usually involves a thorough assessment of the operational (need), financial and technical aspects of a proposal. Feasibility study is the test of the system proposal made to identify whether the user needs may be satisfied using the current software and hardware technologies, whether the system will be cost effective from a business point of view and whether it can be developed with the given budgetary constraints. A feasibility study should be relatively cheap and done at the earliest possible time. Depending on the study, the decision is made whether to go ahead with a more detailed analysis.

All projects are feasible when given unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies.

The study will decide if the proposed system will be cost effective from the business point of view and if it can be developed in the given existing budgetary constraints. The feasibility study should be relatively cheap and quick. The result should inform the decision of whether to go ahead with a more detailed analysis.

Feasibility study may be documented as a separated report to higher officials of the top-level management and can be included as an appendix to the system specification. Feasibility and risk analysis is related in many ways. If there is more project risk then the feasibility of producing the quality software is reduced.

When a new project is proposed, it normally goes through feasibility assessment. Feasibility study is carried out to determine whether the proposed system is possible to develop with available resources and what should be the cost consideration. Facts considered in the feasibility analysis were.

* Operational feasibility
* Technical feasibility
* Economical feasibility
* Behavioural feasibility
* Software feasibility
* Hardware feasibility

**Operational Feasibility**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organization's operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management?
* Are current business methods acceptable to the users?
* Have the users been involved in the planning and development of the project?
* Will the proposed system cause harm?

The purpose of the operational feasibility study is to determine whether the new system will be used if it is developed and implemented. And whether there will be resistance from users that will undermine the possible application benefits. There was no difficulty in, implementing the system and the proposed system is so effective, user friendly and functionally reliable so that the users in the company will find that the new system reduce their hard-steps. If the users of the system are fully aware of the internal working of the system then the users will not be facing any problem in running the system.

**Technical Feasibility**

Technical feasibility study checks the new system is technically very efficient. Also used for now the development of the system in the new technology or not. This involves financial consideration to accommodate technical enhancement.

A study of function, performance and constraints may improve the ability to create an acceptable system. Technical feasibility is frequently the most difficult area to achieve at the stage of product engineering process.

Considering that are normally associated with the technical feasibility include

* Development risk
* Resource availability
* Technology

Technical feasibility study deals with the hardware as well as software requirements. The scope was whether the work for the project is done with the current equipments and the existing software technology has to be examined in the feasibility study. The outcome was found to be positive. In the proposed system, data can be easily stored and managed using database management system software. The reports and results for various queries can be generated easily. Therefore, the system is technically feasible.

**Economical Feasibility**

A cost evaluation is weighed against the ultimate income or benefit derived from the developed system or product. Economic j1.lstification is generally the "bottom-line" consideration that includes cost benefit analysis, long term corporate income strategies, impact on other profit centres or products, cost of resources needed for development and potential market growth. When compared to the advantage obtained from implementing the system its cost is affordable. Also the system is designed to meet the modifications required in the future. So most of the required modifications can be done without much re-work.

Proposed system was developed with the available resources. Since cost input for the software is almost nil the output of the software is always a profit. Hence Software is economically feasible. In the existing system, manpower is more required. In the proposed system, number of employees to be involved is reduced drastically. So, the proposed system is said to be economic. In the existing system, storage of the records should be properly done and security should be provided for the records. In the proposed system, the software provides security and maintenance and it hardly needs one or two persons to operate the system.

**Behavioural Feasibility**

People are inherently resistant to changes and computer is known for facilitating the changes. An estimate should be made of how strongly the user staff reacts towards the developments of the computerized system. In the existing system more manpower is required and time factor is more. In the proposed system, both man power and time factors are reduced and also unnecessary burden is reduced. Thus, the remaining people are made to engage in some other important work. Therefore, the system is behaviourally feasible

**Software Feasibility**

Even though software is developed in a very high software environment, it will be supported by many other platforms and environments with minimum changes.

**Hardware Feasibility**

The software can be developed with resource already existing. Here the consideration is that the existing hardware resources support the technologies that are to be used by the new system. No hardware was newly bought for the project and hence. Software is said to achieve hardware feasibility.

**TECHNOLOGIES USED BEHIND**

The Electric Vehicle Charging Station Finder System is a web-based application that utilizes a variety of technologies to deliver an efficient and effective user experience. Some of the key technologies used in the system are:

1. **HTML**

**Hypertext Mark-up Language** (**HTML**) is the standard [mark-up language](https://en.wikipedia.org/wiki/Markup_language) for documents designed to be displayed in a [web browser](https://en.wikipedia.org/wiki/Web_browser). It can be assisted by technologies such as [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [scripting languages](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript).

[Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [web server](https://en.wikipedia.org/wiki/Web_server) or from local storage and [render](https://en.wikipedia.org/wiki/Browser_engine) the documents into multimedia web pages. HTML describes the structure of a [web page](https://en.wikipedia.org/wiki/Web_page) [semantically](https://en.wikipedia.org/wiki/Semantic_Web) and originally included cues for the appearance of the document.

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) are the building blocks of HTML pages. With HTML constructs, [images](https://en.wikipedia.org/wiki/HTML_element#Images_and_objects) and other objects such as [interactive forms](https://en.wikipedia.org/wiki/Fieldset) may be embedded into the rendered page. HTML provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](https://en.wikipedia.org/wiki/Semantics) for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes and other items. HTML elements are delineated by *tags*, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets). Tags such as <img> and <**input**/> directly introduce content into the page. Other tags such as <**p**> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a [scripting language](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript), which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

1. **CSS**

**Cascading Style Sheets** (**CSS**) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [mark-up language](https://en.wikipedia.org/wiki/Markup_language) such as [HTML](https://en.wikipedia.org/wiki/HTML). CSS is a cornerstone technology of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), alongside HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript).

CSS is designed to enable the separation of presentation and content, including [layout](https://en.wikipedia.org/wiki/Page_layout), colours, and [fonts](https://en.wikipedia.org/wiki/Typeface). This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple [web pages](https://en.wikipedia.org/wiki/Web_page) to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be [cached](https://en.wikipedia.org/wiki/Cache_(computing)) to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same mark-up page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or [screen reader](https://en.wikipedia.org/wiki/Screen_reader)), and on [Braille-based](https://en.wikipedia.org/wiki/Braille_display) tactile devices. CSS also has rules for alternate formatting if the content is accessed on a [mobile device](https://en.wikipedia.org/wiki/Mobile_device).

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

1. **JAVA SCRIPT**

**JavaScript** often abbreviated as **JS**, is a [programming language](https://en.wikipedia.org/wiki/Programming_language) . JavaScript is [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), often [just-in-time compiled](https://en.wikipedia.org/wiki/Just-in-time_compilation), and [multi-paradigm](https://en.wikipedia.org/wiki/Programming_paradigm). It has [curly-bracket syntax](https://en.wikipedia.org/wiki/List_of_programming_languages_by_type#Curly-bracket_languages), [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing), [prototype-based](https://en.wikipedia.org/wiki/Prototype-based_programming) [object-orientation](https://en.wikipedia.org/wiki/Object-oriented_programming), and [first-class functions](https://en.wikipedia.org/wiki/First-class_function).

Alongside [HTML](https://en.wikipedia.org/wiki/HTML) and [CSS](https://en.wikipedia.org/wiki/CSS), JavaScript is one of the core technologies of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web). JavaScript enables interactive [web pages](https://en.wikipedia.org/wiki/Web_page) and is an essential part of [web applications](https://en.wikipedia.org/wiki/Web_application). The vast majority of [websites](https://en.wikipedia.org/wiki/Website) use it for [client-side](https://en.wikipedia.org/wiki/Client-side) page behaviour, and all major [web browsers](https://en.wikipedia.org/wiki/Web_browser) have a dedicated [JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine) to execute it.

As a multi-paradigm language, JavaScript supports [event-driven](https://en.wikipedia.org/wiki/Event-driven_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming), and [imperative](https://en.wikipedia.org/wiki/Imperative_programming) [programming styles](https://en.wikipedia.org/wiki/Programming_paradigm). It has [application programming interfaces](https://en.wikipedia.org/wiki/Application_programming_interface) (APIs) for working with text, dates, [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), standard [data structures](https://en.wikipedia.org/wiki/Data_structure), and the [Document Object Model](https://en.wikipedia.org/wiki/Document_Object_Model) (DOM). However, the language itself does not include any [input/output](https://en.wikipedia.org/wiki/Input/output) (I/O), such as [networking](https://en.wikipedia.org/wiki/Computer_network), [storage](https://en.wikipedia.org/wiki/Data_storage), or [graphics](https://en.wikipedia.org/wiki/Computer_graphics) facilities, as the host environment (usually a web browser) provides those APIs.

JavaScript engines were originally used only in web browsers, but they are now embedded in some [servers](https://en.wikipedia.org/wiki/Server_(computing)), usually via [Node.js](https://en.wikipedia.org/wiki/Node.js). They are also embedded in a variety of applications created with [frameworks](https://en.wikipedia.org/wiki/Software_framework) such as [Electron](https://en.wikipedia.org/wiki/Electron_(software_framework)) and [Cordova](https://en.wikipedia.org/wiki/Apache_Cordova).

Although there are similarities between JavaScript and [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), including language name, [syntax](https://en.wikipedia.org/wiki/Syntax_(programming_languages)), and respective [standard libraries](https://en.wikipedia.org/wiki/Standard_library), the two languages are distinct and differ greatly in design.

1. **Python:**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossumand first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically type dandgarbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

1. **MySQL:**

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage and organize data. MySQL is widely used in academic project development as well as commercial applications due to its flexibility, performance, and scalability. MySQL supports many operating systems, including Linux, Windows, macOS, and Solaris.

MySQL is widely used in academic project development due to its powerful features such as:

1. **Easy to use:** MySQL is easy to learn and use. It has a simple and user-friendly interface that makes it easy for developers to manage and organize data.
2. **Scalability:** MySQL is highly scalable, which means it can handle large amounts of data. It can also be used in distributed systems and can be easily integrated with other technologies.
3. **Performance**: MySQL is known for its high performance and speed. It can handle a large number of transactions per second and can provide quick results for queries.
4. **Security**: MySQL has advanced security features that provide protection against unauthorized access and data breaches. It supports various authentication methods and encryption protocols.
5. **Open source:** MySQL is open-source software, which means it is free to use and can be customized according to the project requirements. It also has a large community of developers who contribute to its development and support.

In academic project development, MySQL is used to store, retrieve, and manage data. It can be used to store various types of data such as text, images, videos, and audio. MySQL is also used in web development frameworks such as Flask, Django, and Ruby on Rails to build dynamic web applications.

In summary, MySQL is a powerful and flexible database management system that is widely used in academic project development due to its scalability, performance, security, and open-source nature.

1. **Flask**

Flask is a Python web framework that allows developers to create web applications quickly and efficiently. It is a lightweight and modular framework that is based on the Werkzeug toolkit and the Jinja2 template engine. Flask provides the user with a number of features such as support for cookies, sessions, and request/response handling. It is also highly extensible, allowing developers to add their own extensions to the framework.

One of the key features of Flask is its simplicity and flexibility. It does not enforce any specific structure or architecture on the developer, allowing them to create their own design patterns and workflows. Flask provides a simple routing system that allows developers to map URL endpoints to Python functions, making it easy to create RESTful APIs.

Another key feature of Flask is its support for Jinja2 templates. Jinja2 is a powerful templating engine that allows developers to write HTML templates using Python syntax. This makes it easy to build dynamic web pages that can be customized based on user input.

Flask also supports the use of blueprints, which allow developers to modularize their code into reusable components. This makes it easy to build large-scale applications that can be easily maintained and extended over time.

In summary, Flask is a powerful and flexible Python web framework that allows developers to build web applications quickly and efficiently. Its simplicity and flexibility make it a popular choice for building a wide range of web applications, including RESTful APIs, e-commerce websites, and content management systems.

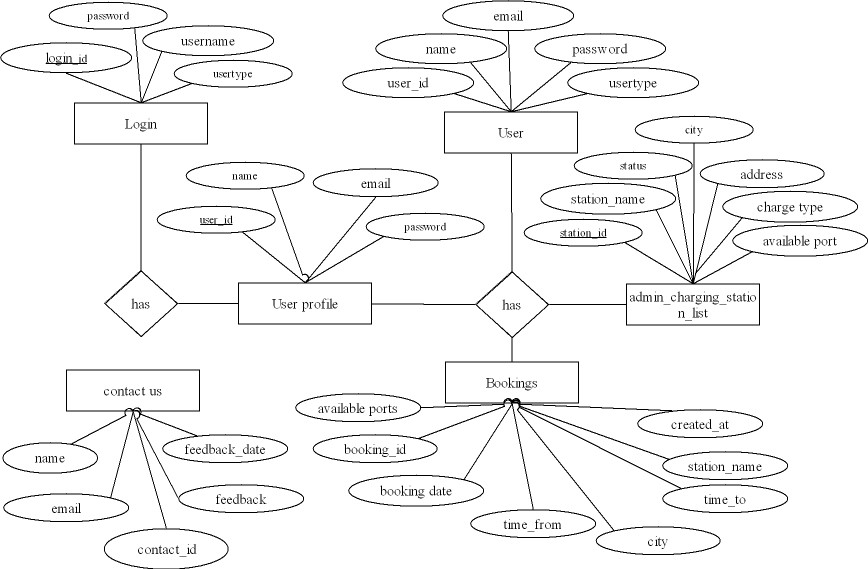
Overall, these technologies work together to create a seamless user experience, allowing users to search for and book charging stations easily and efficiently

**SYSTEM DESIGN**

In the design phase the architecture is established. This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture. The architecture defines the components, their interfaces and behaviours. The deliverable design document is the architecture. The design document describes a plan to implement the requirements. This phase represents the ``how'' phase. Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, and many other details are established.

## ENTITY-RELATIONSHIP DIAGRAM

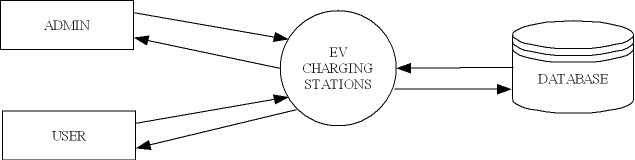
An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. The E-R data model employs three basic notations: entity sets, relationship sets, and attributes.

****

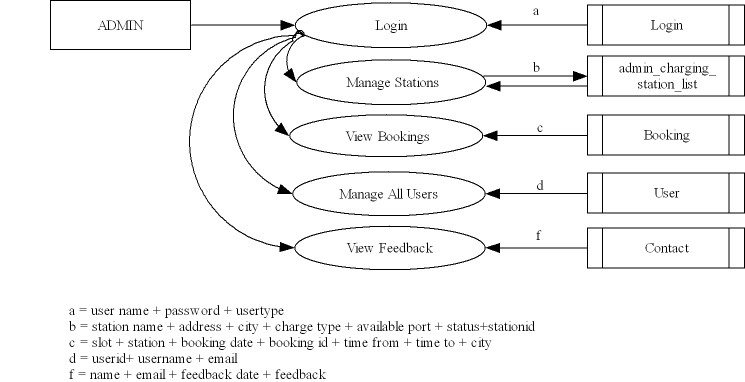
## DATA FLOW DIAGRAM

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored.

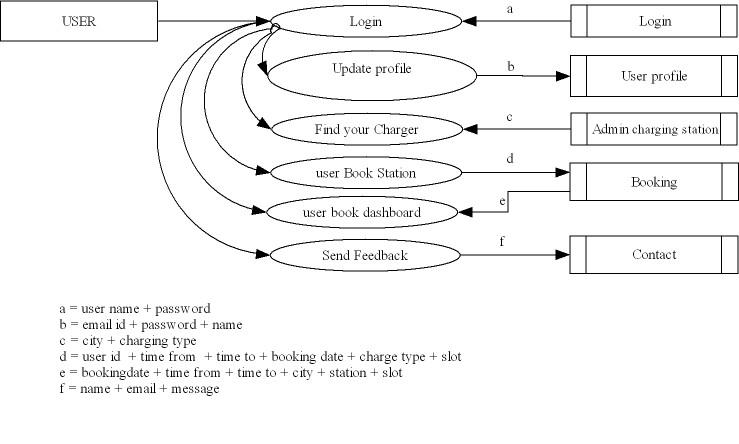
**LEVEL 0**



**LEVEL 1**

****

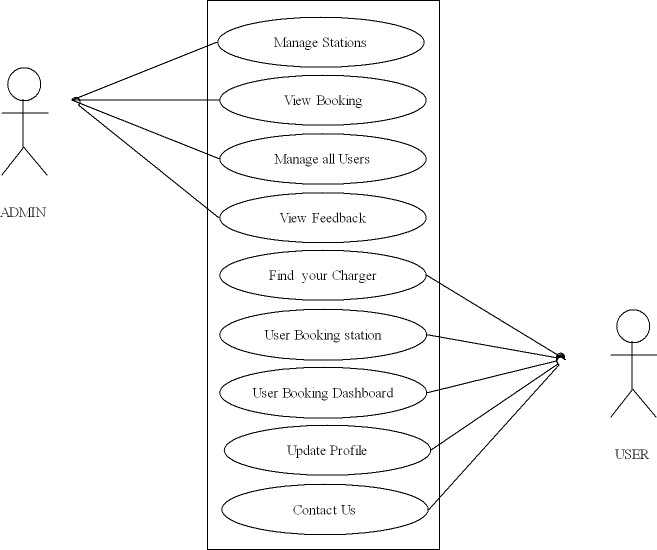
**LEVEL 2**

****

## UML DIAGRAMS

### USECASE DIAGRAM

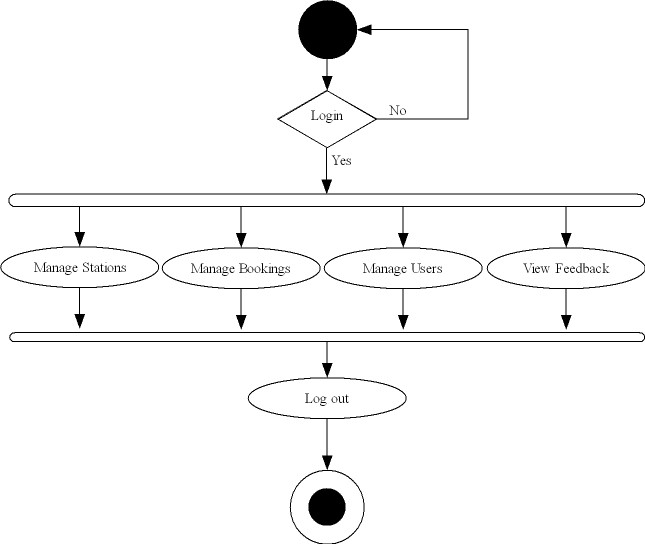
Use Case Diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a use case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use case), and any dependencies between those use case.

****

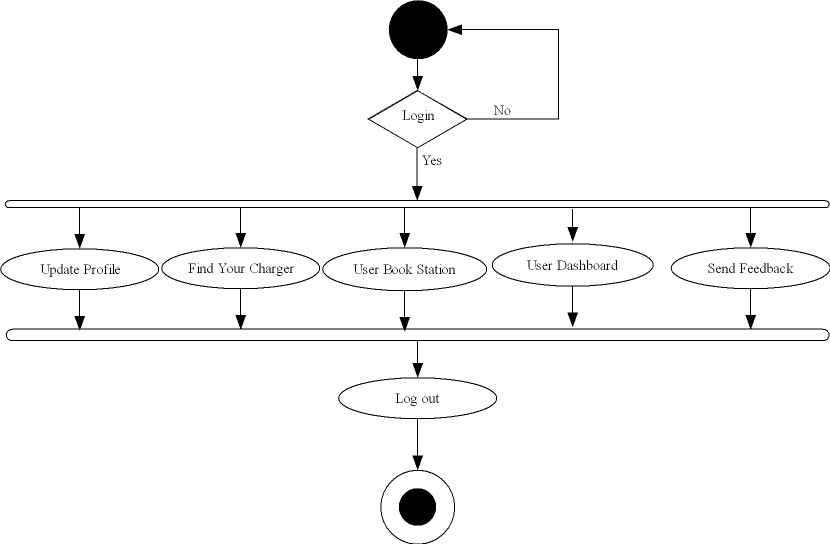
### ACTIVITY DIAGRAM

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modelling. They can also describe the steps in a use case diagram. Activities modelled can be sequential and concurrent. Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

**ADMIN**



**USER**

****

## DATABASE DESIGN

The objective of database design is to provide auxiliary storage and to contribute to the overall efficiency of the computer program component. One auxiliary storage medium must provide efficient access to the data. The general theme behind a database is to handle information as an integrated whole. The general objective is to make information access easy, quick, inexpensive and flexible for the user. In a database environment, common data are available in which several users can use. The concept behind a database is an integrated collection of data and provides a centralized access to the data from the program. It makes possible to treat data as a separate resource.

### 

### TABLES

1. **User\_profile**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| user\_id | Int(11) | Primary key |
| First\_name | Varchar(50) |  |
| Last\_name | Varchar(50) |  |
| Email | Varchar(100) |  |
| Phone\_no | Varchar(20) |  |
| Address | Varchar(200) |  |

1. **User**

|  |  |  |
| --- | --- | --- |
| Field | Type | description |
| User\_id | Int(50) | Primary key |
| Name | Varchar(50) |  |
| Email | Varchar(50) |  |
| User\_type | Varchar(50) |  |

1. **Bookings**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Booking\_id | Int(50) | Primary key |
| Time\_from | time |  |
| Time\_to | time |  |
| Station\_name | varchar(11) |  |
| login\_id | Int(100) | Foreign key |
| Created\_id | timestamp |  |
| City | Varcha(50) |  |
| Booking\_date | Date |  |
| Available\_ports | Varchar(11) |  |

1. **Login**

|  |  |  |
| --- | --- | --- |
| Field | Key | Description |
| Login\_id | Int(11) | Primary key |
| username | Varchar(50) |  |
| Password | Varchar(50) |  |
| usertype | Varchar(50) |  |

1. **Contact\_us**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Sl\_no | Int(50) | Primary key |
| Name | Varchar(50) |  |
| Email | Varchar(50) |  |
| Feedback\_date | Date |  |
| Feedback | Varchar(100) |  |

1. **admin\_charging\_station\_list**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| station\_id | Int(11) | Primary key |
| Station\_name | Varchar(50) |  |
| Address | Varchar(100) |  |
| City | Char(20) |  |
| Charger\_type | Char(20) |  |
| Available\_ports | Varchar(50) |  |
| Status | ENUM |  |

**MODULARITY CRITERIA**

This system contains 2 modules. That are given by

1. **ADMINISTRATION**

* Manage stations
* View bookings
* View all users
* View feedback

1. **USER MANAGEMENT**

* Registration
* Login
* Update profile
* View booking
* Book Station
* Send feedback

**SYSTEM IMPLEMENTATION & TESTING**

**SYSTEM IMPLIMENTATION**

Implementation is the stage in the project where the theoretical design is turned into working system and is given confidence on the new system for users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on the implementation, design of methods to achieve a change over, an evaluation of change over methods. Apart from planning major tasks of preparing the implementation are educating and planning of users. The more complex the system is being implemented, the more involved will be the system analysis and design effort required just for implementation.

The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system.

Implementation is the final and important phase. The most critical stage in achieving a successful new system and in giving the users confidence that the new system will work effective. The system can be implemented only after through testing is done and if it found to working according to the specification. This method also offers the greatest security.

**TESTING**

One of the essential parts of the software development demonstrates the correctness of the software. Software testing is one element of a broader topic that is verification and validation .test are conducted with the intention of finding errors in the software.

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

System Testing is the testing of a complete and fully integrated software product. Usually software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer based system.

Two Category of Software Testing

* Black Box Testing
* White Box Testing
* System test falls under the black box testing category of software testing.

White box testing is the testing of the internal workings or code of a software application. In contrast, black box or System Testing is the opposite. System test involves the external workings of the software from the user's perspective.

System Testing involves testing the software code for following

* Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
* Verify thorough testing of every input in the application to check for desired outputs.
* Testing of the user's experience with the application.
* That is a very basic description of what is involved in system testing. You need¬ to build detailed test cases and test suites that test each aspect of the application as seen from the outside without looking at the actual source code.

**Main Objectives of System testing are:**

As with almost any technical process, software testing has a prescribed order in which things should be done. The following is a list of software testing categories arranged in chronological order. These are the steps taken to fully test new software in preparation for marketing it:

* Unit testing - testing performed on each module or block of code during¬development. Unit Testing is normally done by the programmer who writes the code.
* Integration testing - testing done before, during and after integration of a new module into the main software package. This involves testing of each individual code module. One piece of software can contain several modules which are often created by several different programmers. It is crucial to test each module's effect on the entire program model.
* System testing - testing done by a professional testing agent on the completed software product before it is introduced to the market.
* Acceptance testing - beta testing of the product done by the actual end users.

**UNIT TESTING**

In this each module of the program is tested individually to find out whether the system is work with proper way .in a property designed system each module should have a precise specification and test cases must be defined to check that module meets its specification, unit testing consider each module to be a stand-alone entity, which does not require other system modules to be present process. Unit testing gives stress on the modules independently of one another, to find errors. The errors resulting from the interaction between modules are initially avoided.

In computer programming, unit testing is as software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harness rescan be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

Because some classes may have references to other classes, testing a class can frequently spill over into testing another class. A common example of this is classes that depend on a database: in order to test the class, the tester often writes code that interacts with the database. This is a mistake, because a unit test should usually not go outside of its own class boundary, and especially should not cross such process/network boundaries because this can introduce unacceptable performance problems to the unit test-suite. Crossing such unit boundaries turns unit tests into integration tests, and when such test cases fail, it may be unclear which component is causing the failure. Instead, the software developer should create an abstract interface around the database queries, and then implement that interface with their own mock object. By abstracting this necessary attachment from the code (temporarily reducing the net effective coupling), the independent unit can be more thoroughly tested than may have been previously achieved. This results in a higher-quality unit that is also more maintainable.

**INTEGRATION TESTING**

Integrated testing is a systematic technology for construction program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit tested modules and build the program structure that has been dictated by design. All the modules are combined in advance. The entire program is tested as a whole. In this case the correction is difficult because the isolation of causes is complicated by the vast experience of the entire program. Once these errors are corrected new one appear and the process continues in a seemingly endless loop. In integrated testing, the modules are combined to form cluster. Each cluster is tested using a driver. As integration moves upward, there is a need for separate test drivers. The selection of an integration strategy depends upon software characteristics and sometimes, project schedules. Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Some different types of integration testing are big-bang, mixed (sandwich), risky hardest, top down, and bottom-up. Other Integration Patterns are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration. In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module. Sandwich testing is an approach to combine top down testing with bottom up testing. One limitation to this sort of testing is that any conditions not stated in specified integration tests, outside of the confirmation of the execution of design items, will generally not be tested.

**VALIDATION TESTING**

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfil its intended use when deployed on appropriate environment.

**White box testing**

White box testing is a test case method that uses control structure and procedural design to drive test cases using while box testing methods.

Software engineer can test cases that:

* Exercise all logical decision on their true or false sites.
* Guarantee that all independent paths with a module have been exercised at least once.
* Exercise internal data structure to ensure validity.
* Execute all loops at their operational bounds.

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the expected outputs. This is analogous to testing nodes in a circuit, e.g.in-circuit testing (ICT). White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

White-box testing is a method of testing the application at the level of the source code. These test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error-free environment by examining any fragile code. These white-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is

The ability to know which line of the code is being executed and being able to identify what the correct output should be.

**Black box testing**

Black box testing focuses on the fundamental requirements on software and on input and output of the module. It enables the software engineers 5to derive set of input condition that will truly exercise all functional requirements of a program. Black box testing is rather a contemporary approach that is likely to uncover different class of errors.

It attempts to find out errors in the following:

* Incorrect and missing function
* Performance errors
* Initialization and termination errors

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance. It is sometimes referred to as specification-based testing.

**ACCEPTANCE TESTING**

Acceptance testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery.

**SYSTEM MAINTANENCE**

The results obtained from the evaluation process help the organization to determine whether its information systems are effective and efficient or otherwise. The process of monitoring, evaluating, and modifying of existing information systems to make required or desirable improvements may be termed as System Maintenance. System maintenance is an ongoing activity, which covers a wide variety of activities, including removing program and design errors, updating documentation and test data and updating user support. For the purpose of convenience, maintenance may be categorized into three classes, namely”

**Corrective Maintenance**: This type of maintenance implies removing errors in a program, which might have crept in the system due to faulty design or wrong assumptions. Thus, in corrective maintenance, processing or performance failures are repaired.

**Adaptive Maintenance**: In adaptive maintenance, program functions are changed to enable the information system to satisfy the information needs of the user. This type of maintenance may become necessary because of organizational changes which may include:

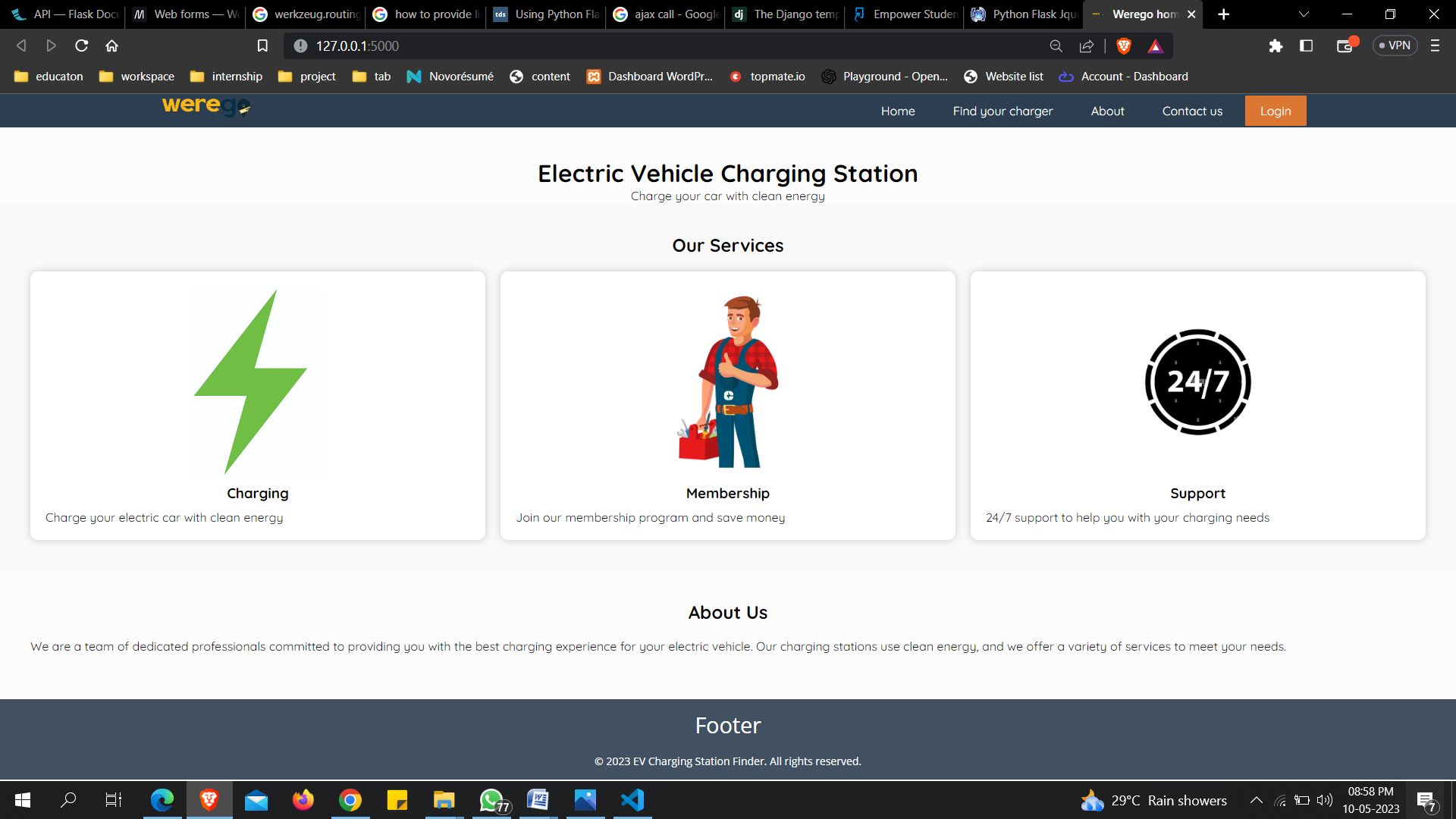
* Change in the organizational procedures.
* Change in organizational objectives, goals, policies, etc.
* Change in forms.
* Change in information needs of managers.
* Change in system controls and security needs, etc

**Perfective Maintenance**: Perfective maintenance means adding new programs or modifying the existing programs to enhance the performance of the information system. This type of maintenance undertaken to respond to user’s additional needs which may be due to the changes within or outside of the organization. Outside changes are primarily environmental changes, which may in the absence of system maintenance; render the information system ineffective and inefficient.

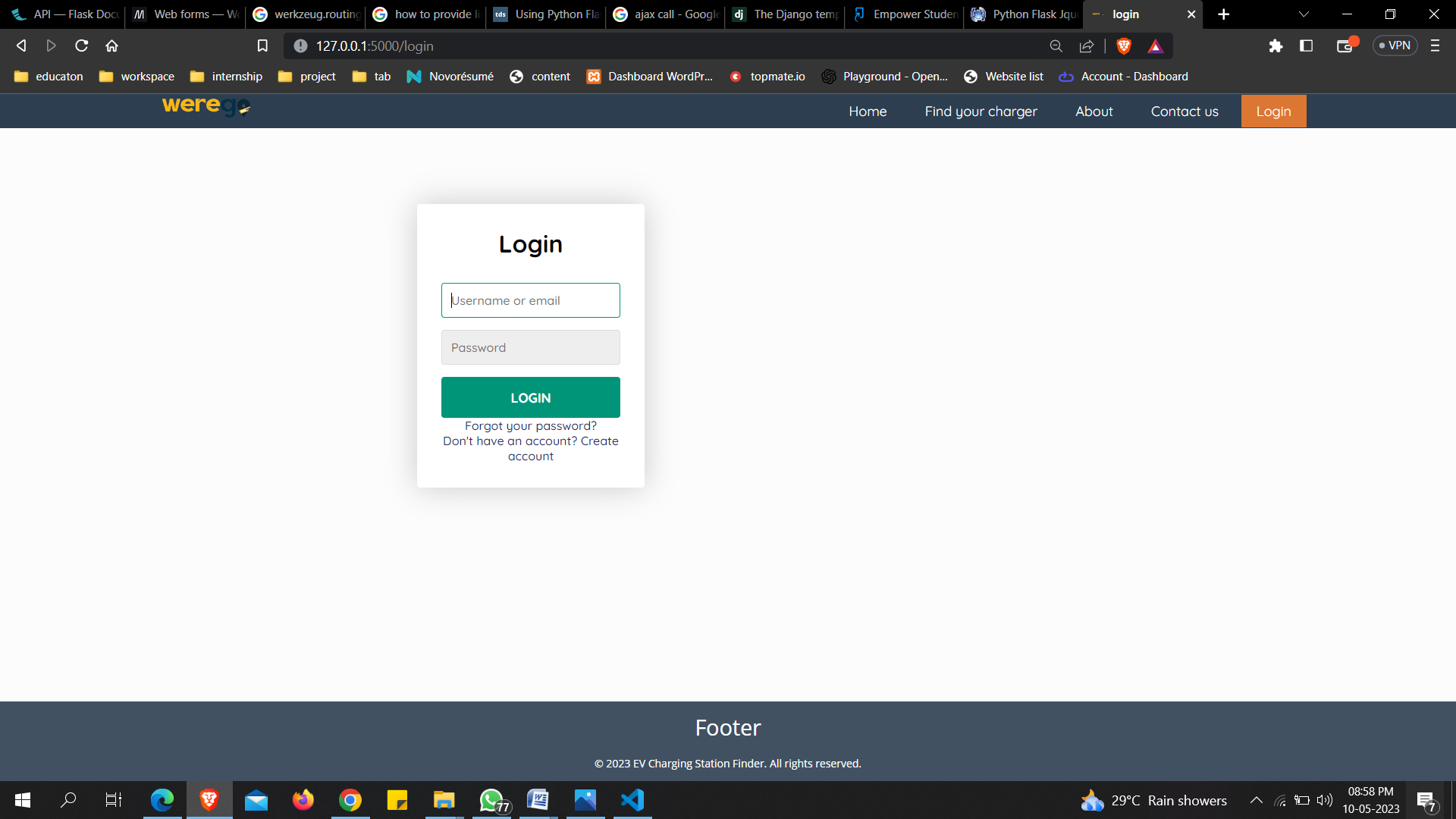
* Changes in governmental policies, laws, etc
* Economic and competitive conditions
* New technology

**SCREENSHOTS**

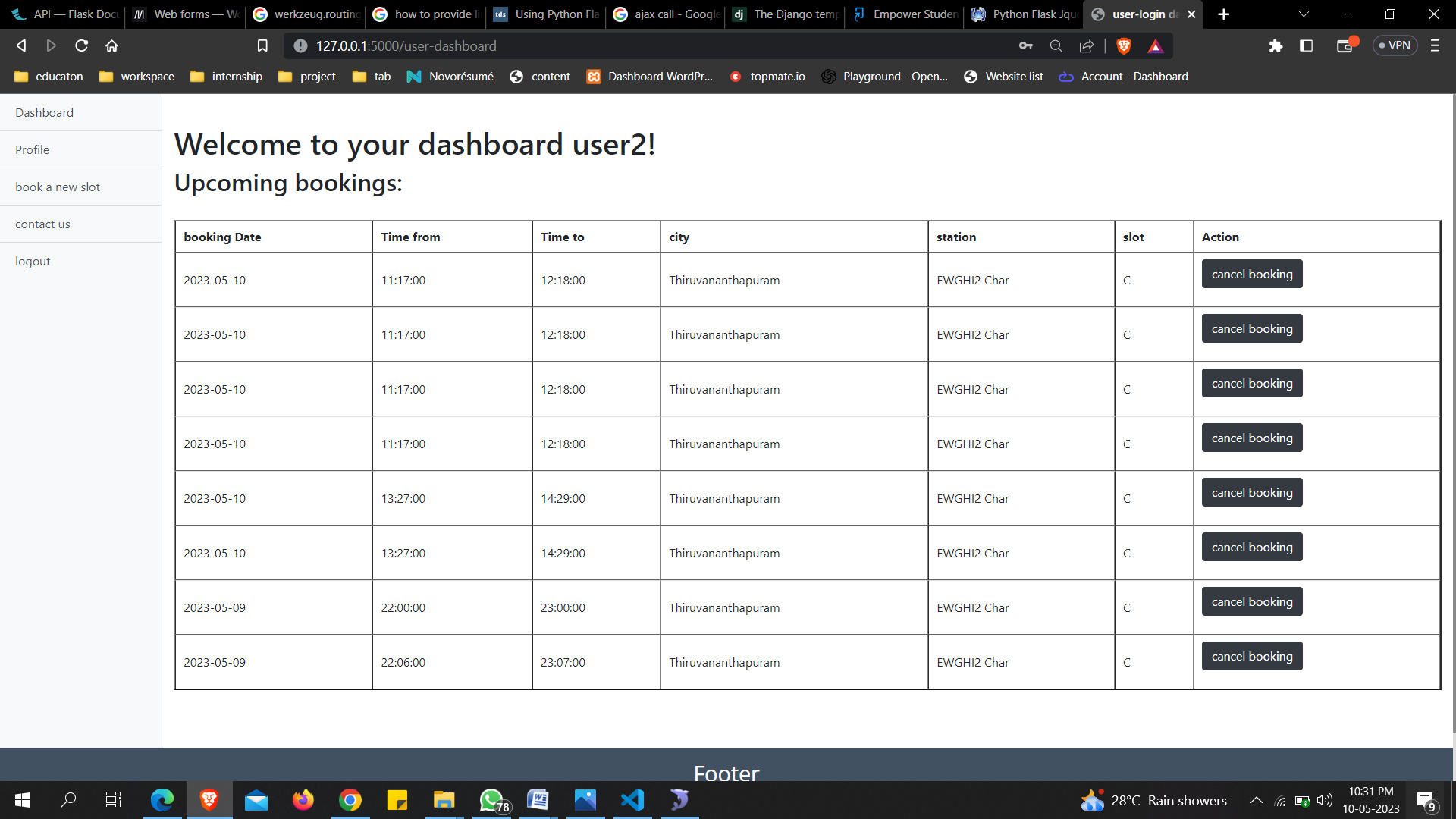
**Home page**

****

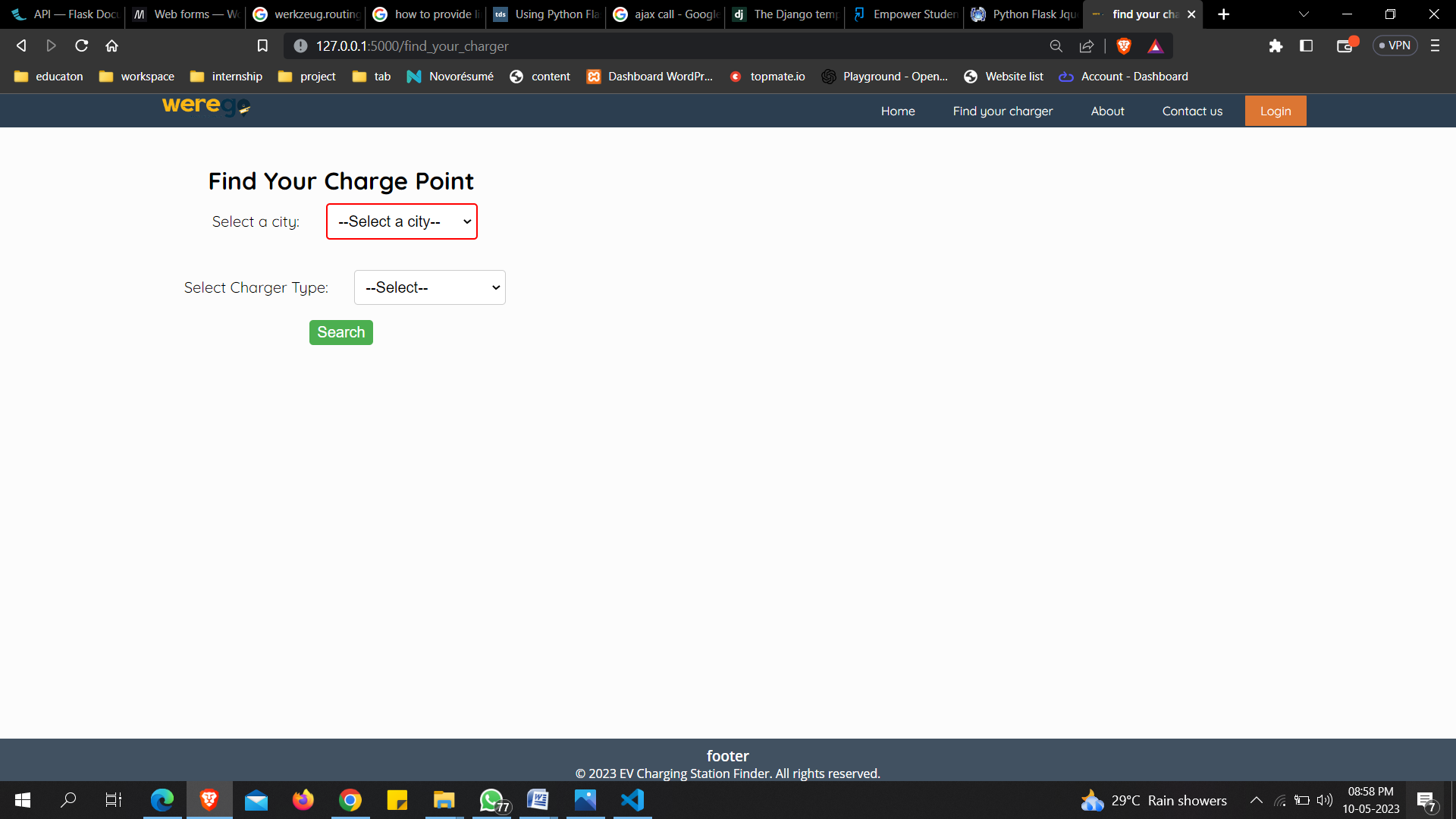
**Login**

****

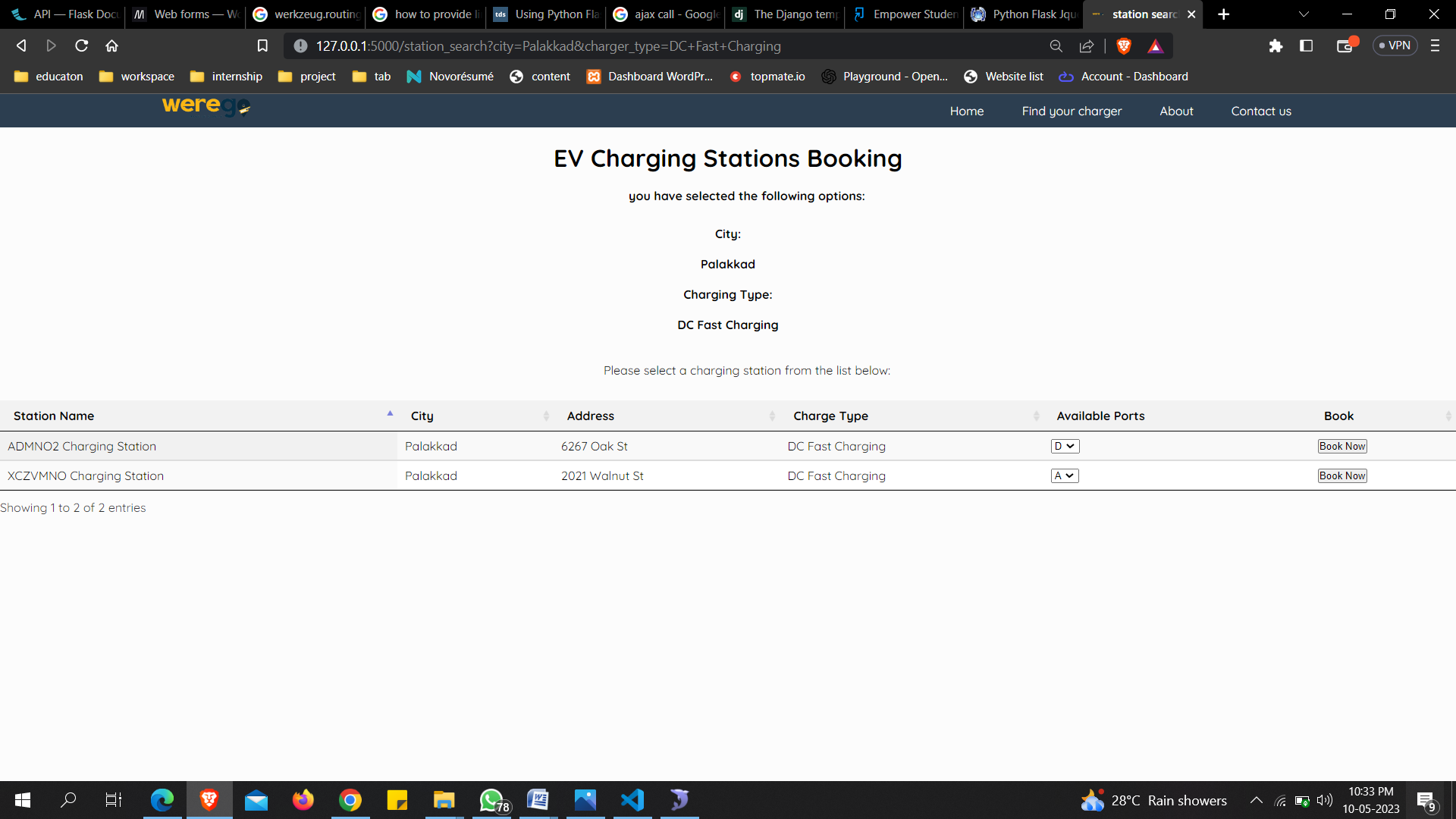
**Home page- user**



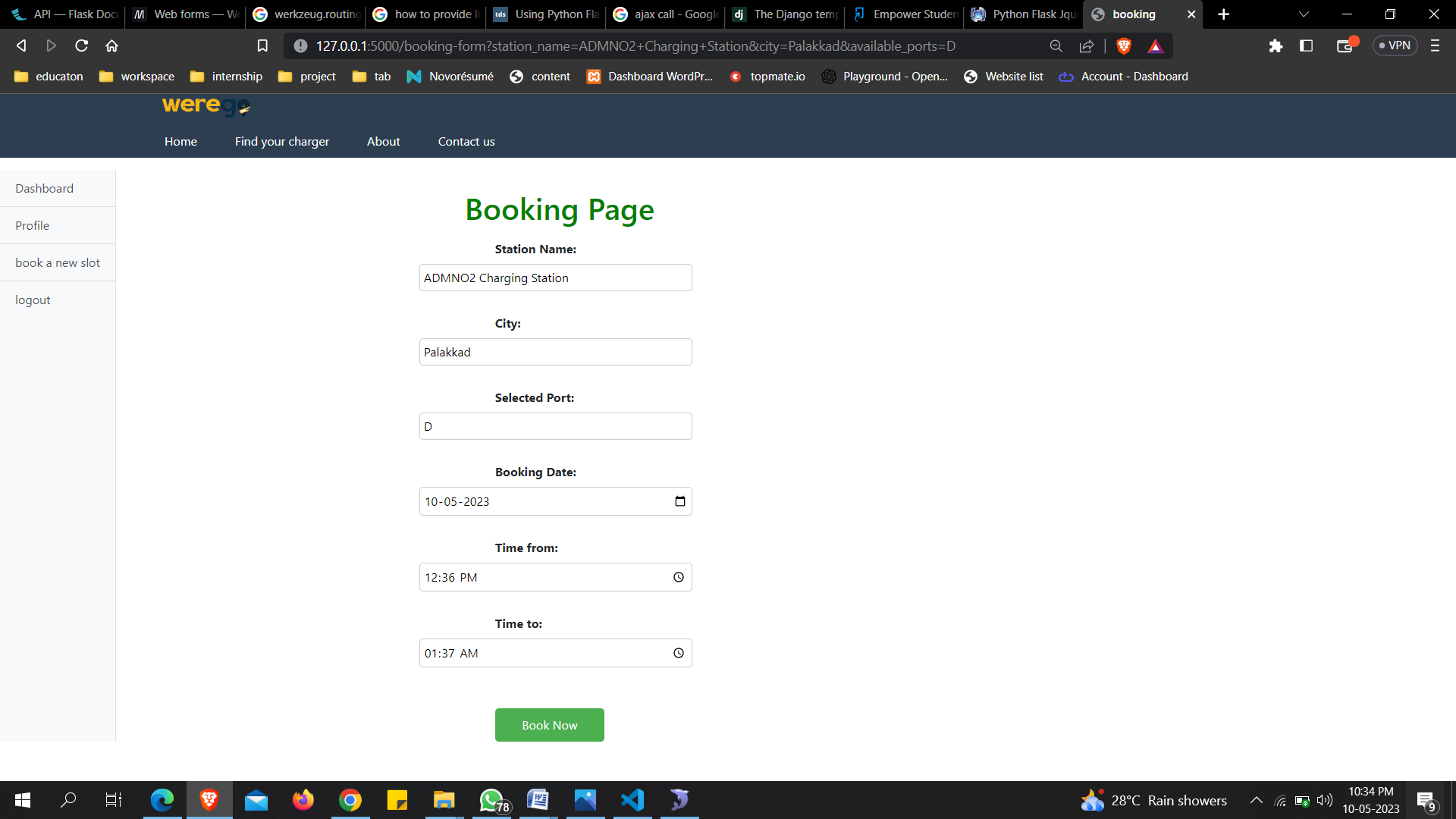
**Find charge- user**



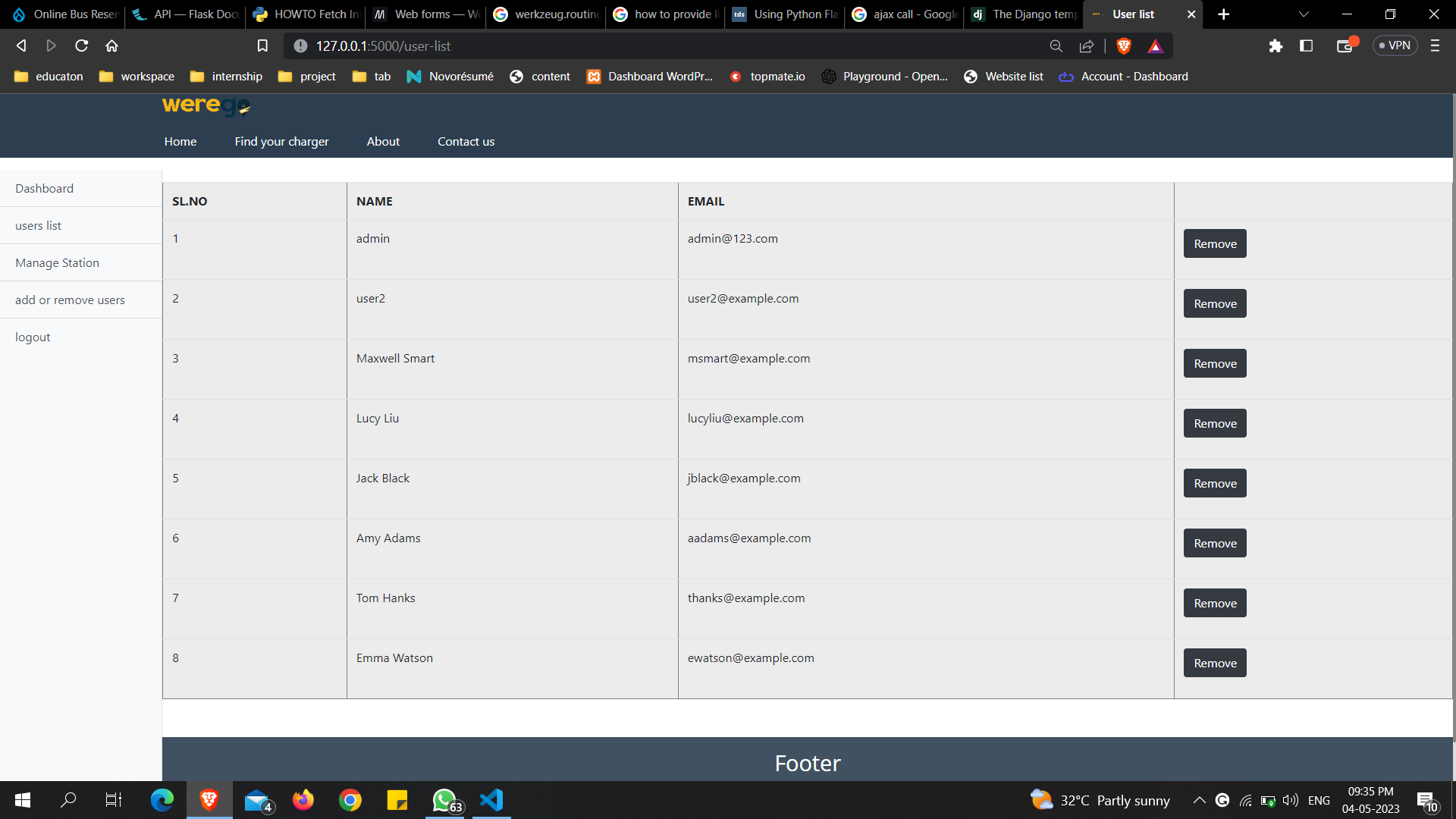
**Station search- user**

****

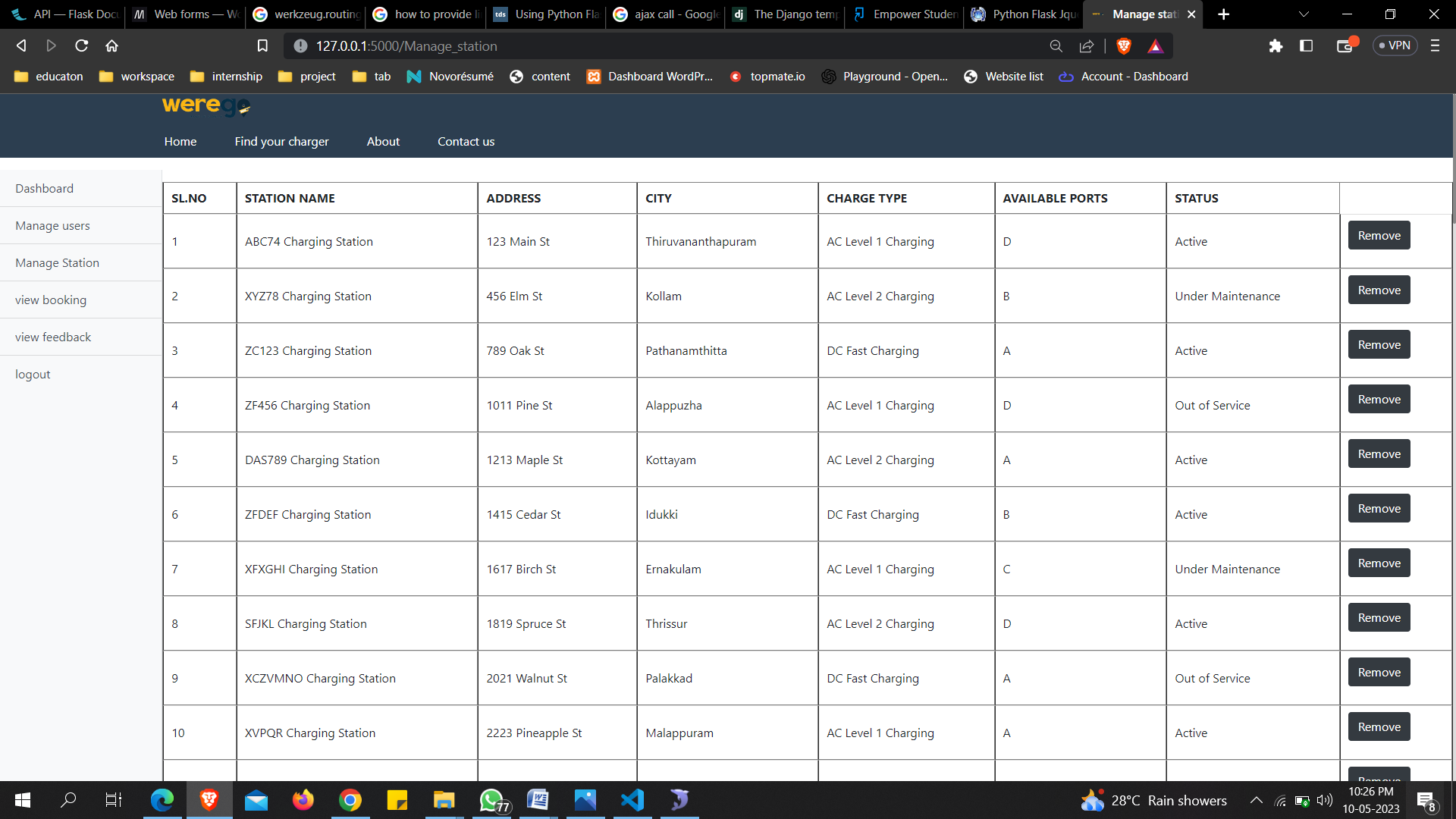
**Booking page-user**



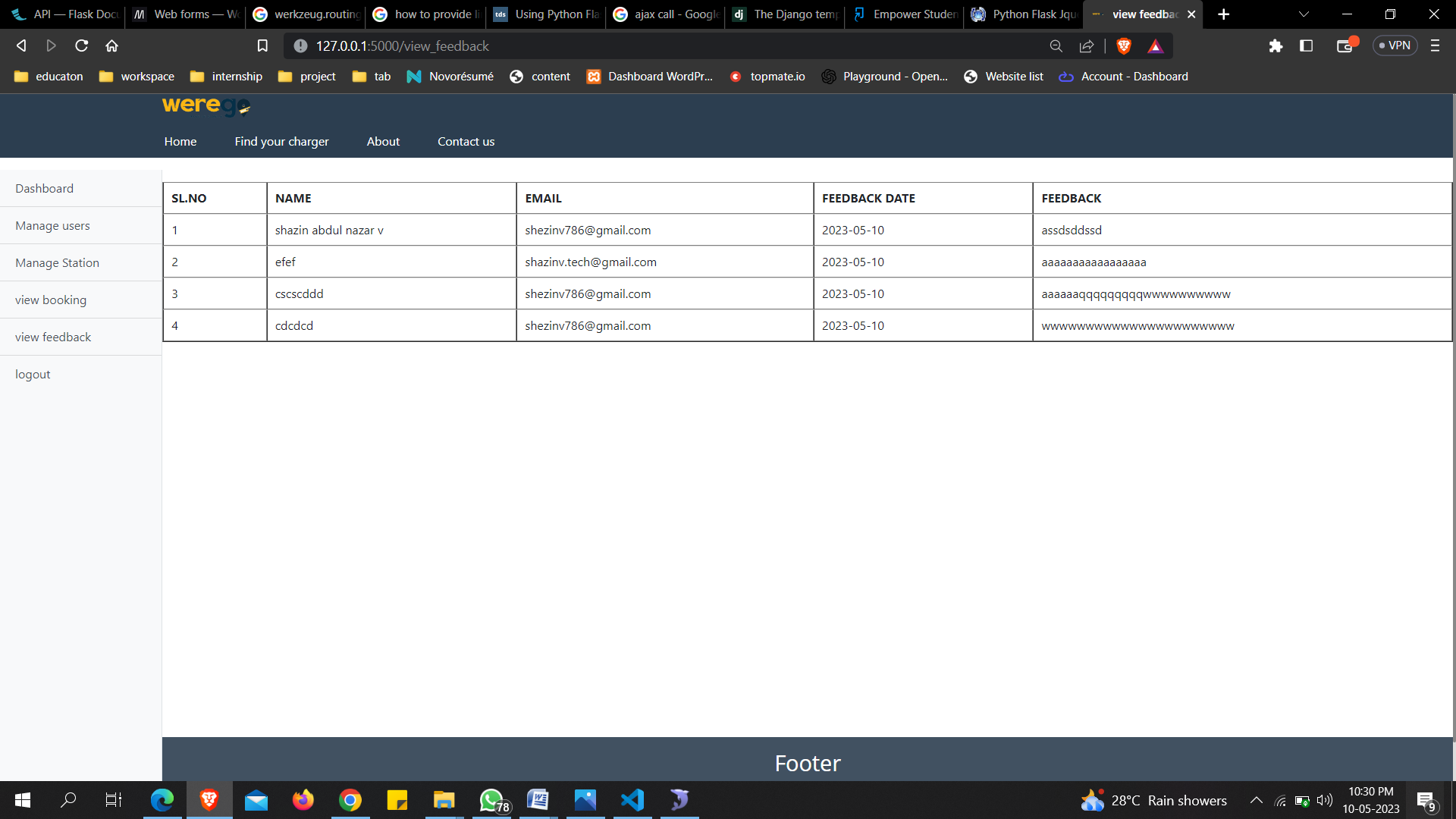
**View Users- Admin**

****

**View Stations- Admin**



**View Feedback- Admin**

****

**Source code**

**Import libraries**

import random

from datetime import datetime

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from flask import Flask, render\_template, url\_for, request, redirect, session, jsonify, flash

from DBConnection import Db

**Find charge.py**

@app.route('/user\_find\_your\_charger', methods=['GET', 'POST'])

def user\_find\_your\_charger():

if 'user\_type' in session and session['user\_type'] == 'user':

        if request.method == 'POST':

            city = request.form.get('city')

            charger\_type = request.form.get('charger\_type')

            db = Db()

            qry = db.select("select station\_name, address, charger\_type, available\_ports from admin\_charging\_station\_list where city = %s and charger\_type = %s", (city, charger\_type))

            return render\_template('user/station\_search.html', data=qry)

        else:

            return render\_template('user/user\_find\_your\_charger.html')

    else:

        return redirect('/')

@app.route('/search\_stations', methods=['POST'])

def search\_stations():

    # Get the form data

    city = request.form.get('city')

    charger\_type = request.form.get('charger\_type')

    # Redirect to the station\_list page with the city and charger\_type as URL parameters

    return redirect(url\_for('station\_search', city=city, charger\_type=charger\_type))

@app.route('/station\_search', methods=['GET'])

def station\_search():

    if 'user\_type' in session and session['user\_type'] == 'user':

        city = request.args.get('city')

        charger\_type = request.args.get('charger\_type')

        # Query your MySQL database using the city and charge\_type variables

        db = Db()

        sql = "select \* from admin\_charging\_station\_list where city = %s and charger\_type = %s"

        ss = db.select(sql, (city, charger\_type))

        # Return the results to the user in a new template

        return render\_template('user/station\_search.html', data=ss, city=city, charger\_type=charger\_type)

    else:

        return redirect('/')

**Booking\_form.py**

@app.route('/booking', methods=['GET', 'POST'])

def booking():

    if request.method == 'POST':

        station\_name = request.form['station\_name']

        city = request.form['city']

        available\_ports = request.form['available\_ports']

        return redirect(url\_for('booking\_form',  station\_name=station\_name, city=city, available\_ports=available\_ports))

    else:

        # handle GET request to display the form

        station\_name = request.args.get('station\_name')

        city = request.args.get('city')

        available\_ports = request.args.get('available\_ports')

        return redirect(url\_for('booking\_form', station\_name=station\_name, city=city, available\_ports=available\_ports))

@app.route('/booking-form', methods=['GET'])

def booking\_form():

    city = request.args.get('city')

    available\_ports = request.args.get('available\_ports')

    station\_name = request.args.get('station\_name')

    db = Db()

    station\_data = db.select("select \* from admin\_charging\_station\_list where station\_name = %s", (station\_name,))

    session['station\_data'] = station\_data[0] if station\_data else None

    if 'station\_data' in session and session['station\_data']:

        return render\_template('/user/booking\_form.html', city=city, available\_ports=available\_ports)

    else:

        return redirect(url\_for('station\_search'))

# from booking to dashboard

@app.route('/book', methods=['POST'])

def book():

    if 'user\_type' in session and session['user\_type'] == 'user':

        # get the form data submitted by the user

        station\_name = request.form['station\_name']

        city = request.form['city']

        available\_ports = request.form['available\_ports']

        booking\_date = request.form['booking\_date']

        time\_from = request.form['time\_from']

        time\_to = request.form['time\_to']

        login\_id = session['uid']

        db = Db()

        # get the current timestamp

        created\_at = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        # insert the booking data into the MySQL table

        sql = "insert into bookings (station\_name, city, available\_ports, booking\_date, time\_from, time\_to, created\_at, login\_id) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"

        booking\_id = db.insert(sql, (station\_name, city, available\_ports, booking\_date, time\_from, time\_to, created\_at, login\_id))

        # redirect the user to their dashboard

        return render\_template("user/user-login-dashboard.html", data={

            'station\_name': station\_name,

            'city': city,

            'available\_ports': available\_ports,

            'booking\_date': booking\_date,

            'time\_from': time\_from,

            'time\_to': time\_to,

            'created\_at': created\_at,

            'booking\_id': booking\_id

        })

    else:

        return redirect('/booking-form')

**Station\_search.html**

<body>

    <h1>EV Charging Stations Booking</h1>

    <p><strong>you have selected the following options:<br><br></strong></p>

    <ul>

        <li><strong>City:<br><br> {{ city }} </strong></li><br>

        <li><strong>Charging Type:<br><br> {{ charger\_type }} </strong></li><br>

      </ul>

    <p>Please select a charging station from the list below:<br><br></p>

    <table id="myTable" class="display">

        <thead>

            <tr>

              <th>Station Name</th>

              <th>City</th>

              <th>Address</th>

              <th>Charge Type</th>

              <th>Available Ports</th>

              <th style="width: 10%" orderable="false">Book</th>

            </tr>

          </thead>

          <tbody>

            {% for station in data %}

            <tr>

              <td>{{ station.station\_name }}</td>

              <td>{{ station.city }}</td>

              <td>{{ station.address }}</td>

              <td>{{ station.charger\_type }}</td>

              <td>

                <select name="ports">

                  {% for port in station.available\_ports %}

                  <option value="{{ port }}">{{ port }}</option>

                  {% endfor %}

                </select>

              </td>

              <td>

                <button class="book-now-btn" data-station="{{ station.station\_name }}" data-city="{{ station.city }}" data-available-ports="{{ station.available\_ports }}">Book Now</button>

              </td>

            </tr>

            {% endfor %}

          </tbody>

    </table>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script src="https://cdn.datatables.net/1.11.4/js/jquery.dataTables.min.js"></script>

<script>

    $(function() {

        $('.book-now-btn').on('click', function() {

            var stationName = $(this).data('station');

            var city = $(this).data('city');

            var available\_ports = $(this).data('available-ports');

            $.ajax({

                url: '/booking',

                method: 'POST',

                data: {

                    station\_name: stationName,

                    city: city,

                    available\_ports: available\_ports

                },

                success: function(response) {

                    window.location.href = '/booking?station\_name=' + stationName + '&city=' + city + '&available\_ports=' + available\_ports;

                },

                error: function(error) {

                console.log("There was an error in ajax: " + error);

}

            });

        });

        $('#myTable').DataTable({

            paging: false,

            searching: false,

            columnDefs: [{ orderable: false, targets: 4 }]

        });

    });

</script>

</body>

**Booking\_form.html**

<div>

  <h1>Booking Page</h1>

  <form action="/book" method="POST">

    <label for="station-name">Station Name:</label><br>

    <input type="text" id="station-name" name="station\_name" value="{{ request.args.get('station\_name') }}" readonly>

    <label for="city">City:</label><br>

    <input type="text" id="city" name="city" value="{{ request.args.get('city') }}" readonly>

    <label for="available\_ports">Selected Port:</label><br>

    <input type="text" id="available\_ports" name="available\_ports" value="{{ request.args.get('available\_ports') }}" readonly>

    <label for="booking\_date">Booking Date:</label><br>

    <input type="date" id="booking\_date" name="booking\_date" required>

    <label for="time\_from">Time from:</label><br>

    <input type="time" id="time\_from" name="time\_from" required>

    <label for="time\_to">Time to:</label>

    <input type="time" id="time\_to" name="time\_to" required><br>

    <button type="submit" onclick="bookNow()">Book Now</button>

  </form>

            <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

            <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"></script>

            <script>

              $(document).ready(function() {

                $(function() {

                  $("#my\_date\_picker1").datepicker({

                    minDate: 0,

                    maxDate: '+1Y',

                    dateFormat: 'dd-mm-yy'

                  });

                });

                $('#my\_date\_picker1').change(function() {

                  startDate = $(this).datepicker('getDate');

                  // Only set minDate if an end date picker exists

                  if ($("#my\_date\_picker2").length) {

                    $("#my\_date\_picker2").datepicker("option", "minDate", startDate);

                  }

                });

                function today() {

                  var d = new Date();

                  return d.toLocaleDateString();

                }

                function now() {

                  var d = new Date();

                  var curr\_hour = d.getHours();

                  var curr\_min = d.getMinutes();

                  // Add leading zeroes to the minute value if it is less than 10

                  if (curr\_min < 10) {

                    curr\_min = "0" + curr\_min;

                  }

                  return (curr\_hour + ":" + curr\_min);

                }

                $('#time\_from').change(function() {

                  startTime = $(this).val();

                  $("#time\_to").attr("min", startTime);

                });

              });

              function bookNow() {

                // Get the values of all the inputs

                const available\_port = $("#available\_ports option:selected").val();

                const booking\_date = $("#my\_date\_picker1").val();

                const time\_from = $("#time\_from").val();

                const time\_to = $("#time\_to").val();

                // Check if any of the fields are empty

                if (available\_port === "" || booking\_date === "" || time\_from === "" || time\_to === "") {

                  if (booking\_date === "") {

                    alert("Please select a date to book.");

                  } else if (time\_from === "" || time\_to === "") {

                    alert("Please select a time to book.");

                  } else {

                    alert("Please fill in all the fields before booking.");

                  }

                } else {

                  // Show a confirmation popup

                  if (confirm("Are you sure you want to book this port?")) {

                    alert("Booking Successful!");

                    // Redirect to user dashboard page

                    window.location.href = "/user-dashboard";

                    // Reload the page after 2 seconds

                    setTimeout(function() {

                      location.reload();

                    }, 2000);

                  }

                }

              }

            </script>

            </div>

**CONCLUSION**

In conclusion, the Electric Vehicle Charging Station Finder System is a solution to the increasing demand for electric vehicle charging stations. This system provides a user-friendly website that allows electric vehicle owners to easily locate and book charging stations. The feasibility study has shown that the system is financially, technically, and operationally feasible.

The system has been designed using modern technologies such as HTML, CSS, JavaScript, Python, and MySQL. These technologies allow for the creation of a responsive and dynamic website that can handle a large amount of data.

Overall, the Electric Vehicle Charging Station Finder System is a valuable tool for electric vehicle owners, charging station operators, and other stakeholders in the electric vehicle industry. It promotes the adoption of electric vehicles by making it easier and more convenient for owners to find and use charging stations.

**FUTURE ENHANCEMENT**

This project has a very vast scope in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion.

The field of EV Charging Station and Slot Booking System still has a lot of issues that are not completely resolved.

Recommendations for the further improvement of this project include:

1. **Notifications:** Implement email or SMS notifications to inform users about their booking status, charging progress, and any updates or changes related to their slot. This will enhance user communication and provide real-time updates.
2. **Bookmark Feature:** Introduce a bookmark feature that allows frequent users to save their preferred charging stations for quick and easy access. This feature will streamline the booking process and improve user convenience.
3. **Geolocation Integration:** Replace dropdown menus with geolocation services to enable users to navigate through the available charging stations based on their current location. This will simplify the station selection process and enhance user experience.
4. **Nearest Station Locator:** Utilize the user's location information to automatically identify and suggest the nearest charging station without the need for manual selection. This feature will save time and effort for users, especially in unfamiliar areas.
5. **Account Verification:** Implement an email notification-based account verification process to enhance system security and ensure that users provide valid and accurate information during registration. This will help maintain the integrity of the system.
6. **Advanced Payment Gateway:** Integrate a secure payment gateway that allows users to make slot payments in advance, eliminating the need for on-site payment and reducing queue times. This feature will enhance user convenience and streamline the charging process.

There can be many more future Enhancement & improvement in the EV Charging Station and Slot Booking System

**BIBILOGRAPHY**

* <https://www.tutorialspoint.com/django/>
* <https://www.w3schools.com/>
* <https://www.python.org/>
* <https://www.geeksforgeeks.org/vgg-16-cnn-model/>
* <https://medium.com/>
* [https://codepen.io](https://codepen.io/)
* [https://codewithrandom.com](https://codewithrandom.com/)
* <https://developer.mozilla.org/en-US>
* <https://overapi.com/css>