CHAPTER 1
INTRODUCTION

The goal of this application is to build a platform for booking automobile services online. It makes it easier for mechanics to list all of the services they provide. Customers have the option of booking one or more services based on their needs. They may sort mechanics based on their location, requirements, and availability via the application.

1.1 PROJECT OVERVIEW

The main goal of this application is to create a platform for online based vehicle service booking as simple as possible by providing a web interface for the mechanics and customer.Mechanics can login to their account and list all the services they offer.

Customers can log in to their account and book one or more services based on their requirements. They can sort mechanics based on factors such as location, car type, mechanic availability, and so on. The mechanics have the ability to accept or deny the booking, as well as alter the status of the scheduled service.

1.2 SYSTEM SPECIFICATIONS

A software requirements specification is a description of a software system to be developed. It is modeled after stakeholder requirements specification.

1.2.1 Minimum Hardware Requirements

PROCESSOR	1 CPU Core
RAM	1 GB
HARD DISK	500 MB

1.2.2 Minimum Software Requirements

OPERATING SYSTEM	Windows | Linux | Mac
IDE	Visual Studio Code | Sublime Text 3
BROWSER	Google Chrome Version 65+
SOFTWARE	Python Pip Virtualenv Django DB Browser for SQLite
DATABASE	SQLite 3

1.3 TECHNOLOGY OVERVIEW

The world of global communications is evolving as a result of technological advancements. This chapter goes into the tools and technologies that were used to create the web application.

1.3.1 HTML

The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.

HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets.

1.3.2 CSS

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages 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 to improve the page load speed between the pages that share the file and its formatting.

1.3.3 BOOTSTRAP

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. Bootstrap is a HTML, CSS & JS Library that focuses on simplifying the development of informative web pages (as opposed to web apps).

The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents.

1.3.4 JAVASCRIPT

JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

1.3.5 JQUERY

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications.

jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

1.3.6 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Django was invented to meet fast-moving newsroom deadlines, while satisfying the tough requirements of experienced Web developers. Django can take Web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of Web development. It’s free and open source.

Security is a topic of paramount importance in the development of Web applications and Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords. It is based on MVT (Model-View-Template) Architecture as shown in Figure 2.1.

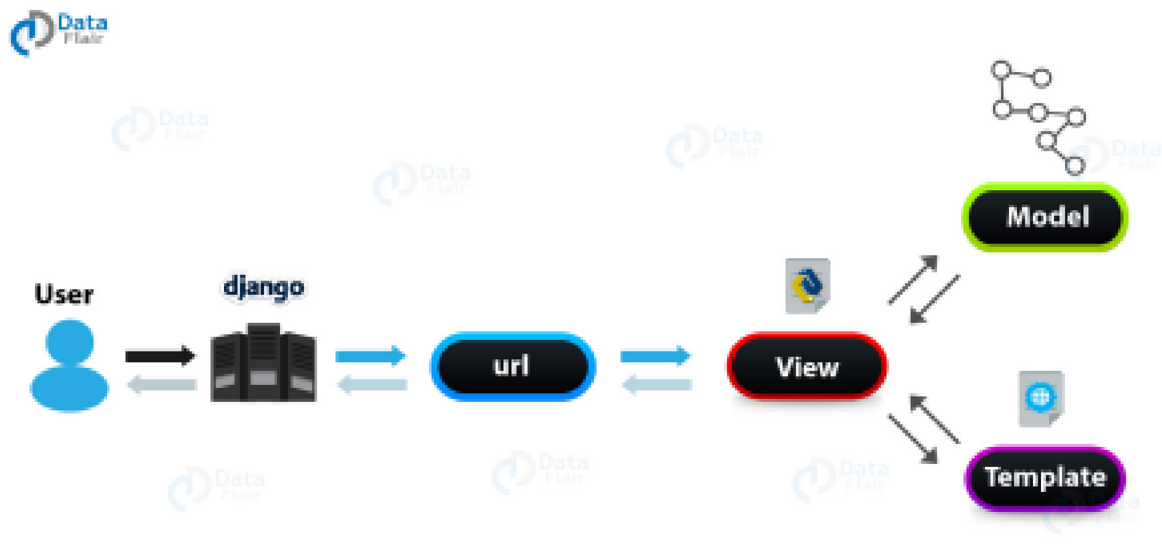


Figure 1.1 Django MVT Architecture

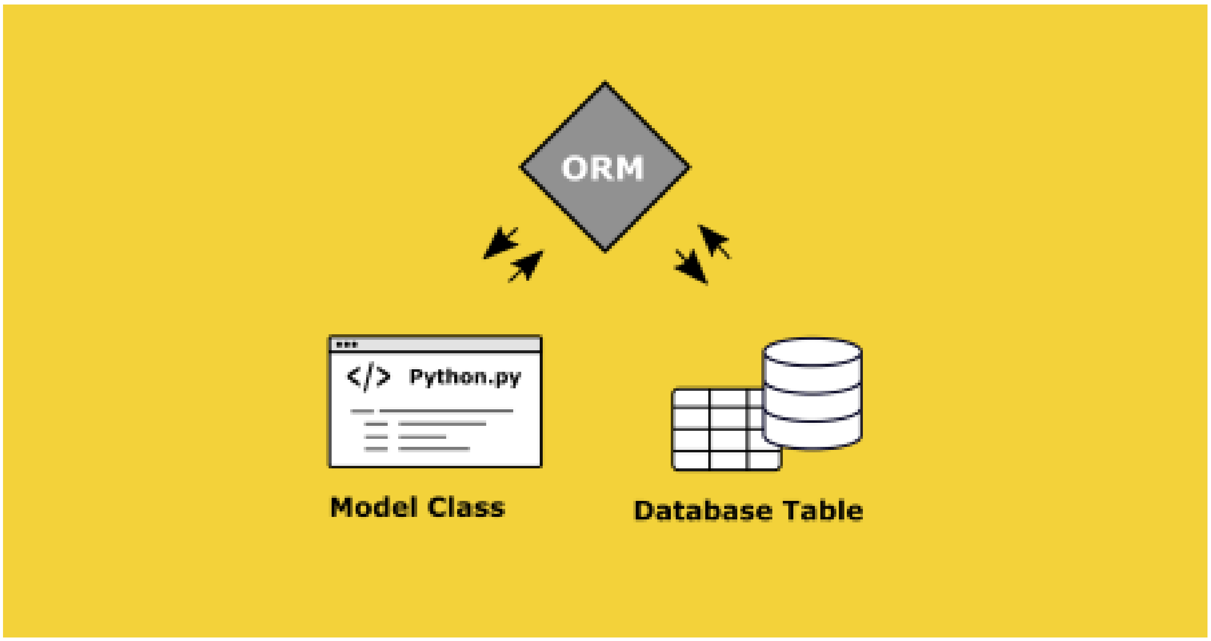
Model Layer

Django provides an abstraction layer (the "models") for structuring and manipulating the data of the Web application. Model is going to act as the interface of the data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database shown in Figure 2.2.

A Django model is the built-in feature that Django uses to create tables, their fields, and various constraints. In short, Django Models is the SQL of Database one uses with Django. SQL (Structured Query Language) is complex and involves a lot of different queries for creating, deleting, updating or any other stuff related to databases. Django models simplify the tasks and organize tables into models. Generally, each model maps to a single database table.

Django models provide simplicity, consistency, version control and advanced metadata handling. Basics of a model include,

- Each model is a Python class that subclasses django.db.models.Model.
- Each attribute of the model represents a database field.
- Django gives an automatically generated database access API.



1.2 Django Model Layer

View Layer

Django has the concept of "views" to encapsulate the logic responsible for processing a user’s request and for returning the response. A View is the user interface. It is represented by HTML/CSS/Javascript files. A view function is a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, anything that a web browser can display. There are two types of views and Figure 2.3 depicts the view and template layer of Django.

Function Based Views

Function based views are written using a function in python which receives as an argument HttpRequest object and returns an HttpResponse Object. Function based views are generally divided into 4 basic strategies, i.e., CRUD (Create, Retrieve, Update, Delete). CRUD is the base of any framework one is using for development.

Class Based Views

Class-based views provide an alternative way to implement views as Python objects instead of functions. They do not replace function-based views, but have certain differences and advantages when compared to function-based views:

- Organization of code related to specific HTTP methods (GET, POST, etc.) can be addressed by separate methods instead of conditional branching.
- Object oriented techniques such as mixins (multiple inheritance) can be used to factor code into reusable components.

Class-based views are simpler and efficient to manage than function-based views. A function-based view with tons of lines of code can be converted into a class-based view with few lines only. This is where Object-Oriented Programming comes into impact.

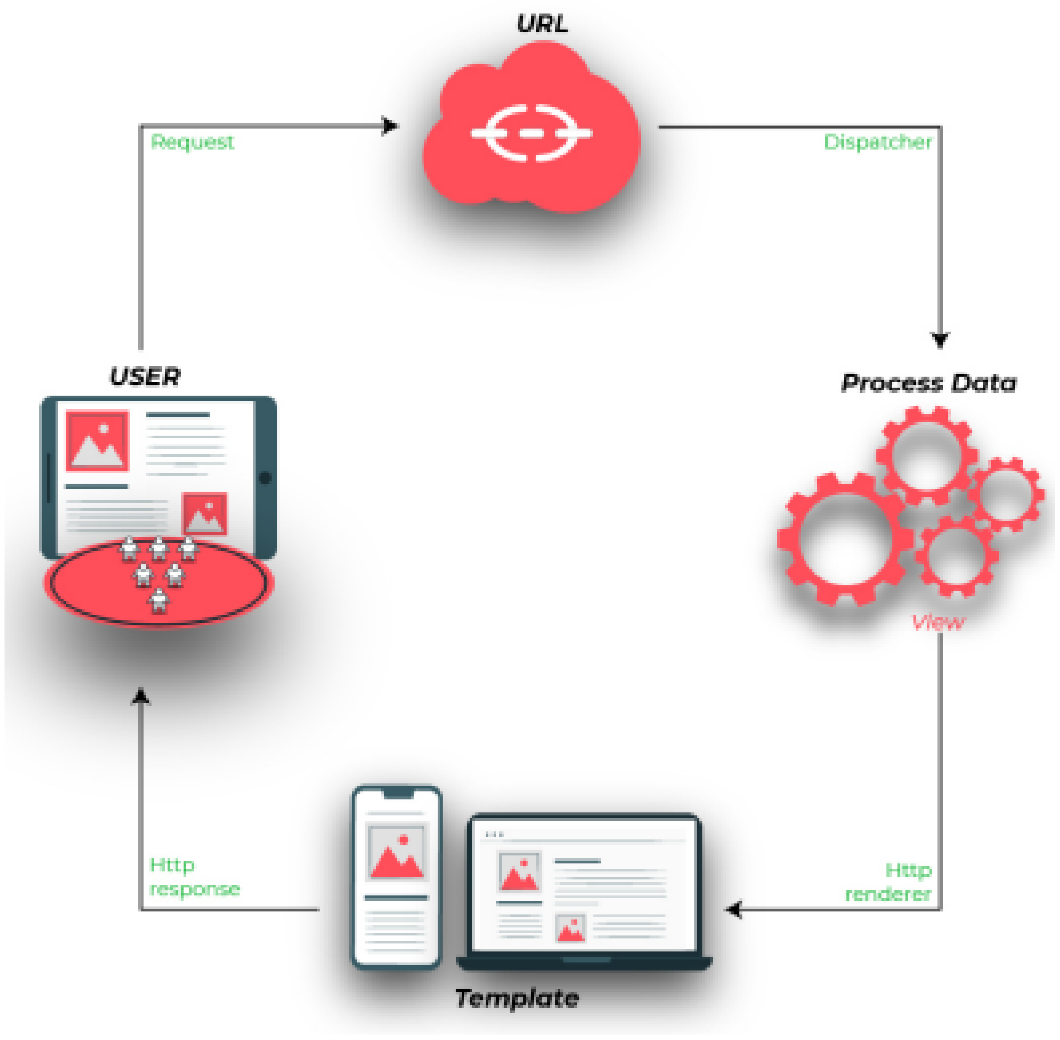


Figure 1.3 Django View and Template Layer

Template Layer

The template layer provides a designer-friendly syntax for rendering the information to be presented to the user. A template in Django is basically written in HTML, CSS and Javascript in an .html file. Django framework efficiently handles and generates dynamically HTML web pages that are visible to the end-user. Django mainly functions with a backend, so in order to provide a frontend and provide a layout to our website, we use templates. There are two methods of adding the template to our website depending on our needs. Single template directory can be used which will spread over the entire project.

Django Template Language

This is one of the most important facilities provided by Django Templates. A Django template is a text document or a Python string marked-up using the Django template language. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags.

The various conditions such as if, else, if-else, empty, etc and also main characteristics of Django Template language like Variables, Tags, Filters and Comments can be used.

Template Inheritance

The most powerful and thus the most complex part of Django's template engine is template inheritance. Template inheritance allows to build a base "skeleton" template that contains all the common elements of the site and defines blocks that child templates can override. extends tag is used for inheritance of templates in Django. One needs to repeat the same code again and again. Extends can be used to inherit templates as well as variables.

1.3.7 SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.SQLite is the most widely deployed database in the world with more applications, including several high-profile projects. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

The database file format is cross-platform - A database may be readily copied between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen().

SQLite is a compact library. With all features enabled, the library size can be less than 750KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory. Nevertheless, performance is usually quite good even in low-memory environments.

CHAPTER 2
SYSTEM ANALYSIS

Systems analysis is "the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way".

3.1 EXISTING SYSTEM

The present system is a manual one in which users keep books to record information about customers, services sought, and so on.
The shortcomings of the existing system are as follows:

- Customer records are challenging to keep track of.
- To get the past data, more manual hours are required.
- Daily service information must be put into books, which can be tough to keep up with.

3.2 PROPOSED SYSTEM

This application aims to create a platform for online based vehicle service booking.It helps the mechanics to list all the services they offer. Customers can choose one or more services to book based on some criteria. It includes location, vehicle type, availability of mechanics and so on.

3.2.1 Advantages Of Proposed System

- Customers do not have to spend hours looking for mechanics.
- Customers can book their services anywhere at any time.
- The mechanics and customers receive notification based on email when a booking is made or its status is updated.

- It will reduce the work process of the mechanics while the data will be kept securely.
- The mechanics will be able to get valuable insight about their business.

3.3 FUNCTIONAL REQUIREMENTS

Mechanics can

- Create / edit / delete all their services and their details.
- View a list of all bookings (pending, ready for delivery and completed).
- View details of each booking.
- Mark a booking as ready for delivery.
- Mark a booking as completed.
- Receive an email whenever a booking is made.

Customers can

- Register for an account with their email address and mobile number.
- Book a service at a particular date.
- See the status of their booking.
- See all their previous bookings.
- Receive an email as soon as their booking is ready for delivery.

CHAPTER 3
SYSTEM DESIGN

The process of specifying the architecture, product design, components, interfaces, and data for a system in order to meet specific requirements is known as systems design.

3.1 UML DIAGRAM

The UML Diagram of the project is shown in Figure 4.1.

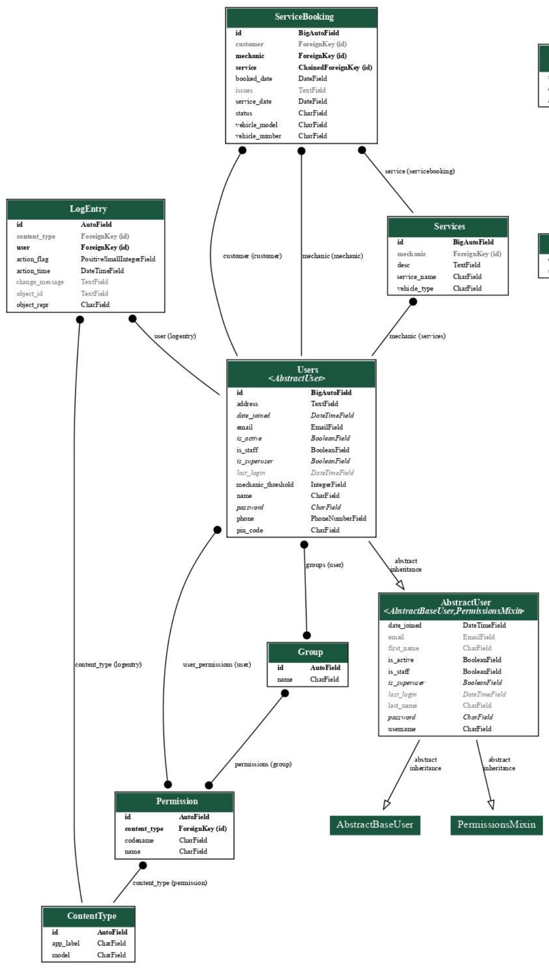


Figure 3.1 UML Diagram

3.2 USE CASE DIAGRAM

The Use Case Diagram of the project is shown in Figure 3.2.

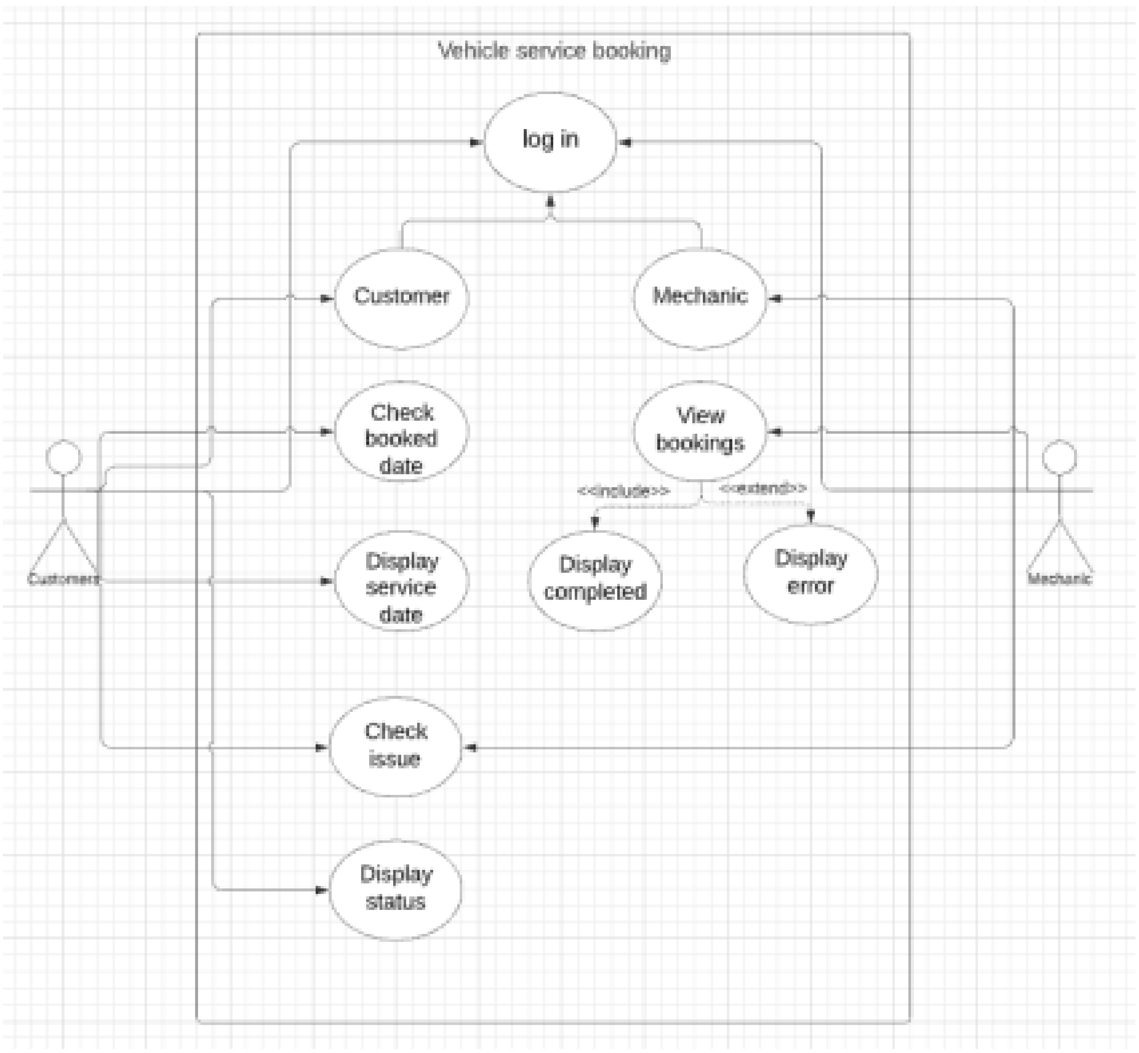


Figure 3.2 Use Case Diagram

3.3 DATA FLOW DIAGRAM

The Data Flow Diagram of the project is shown in Figure 3.3 .

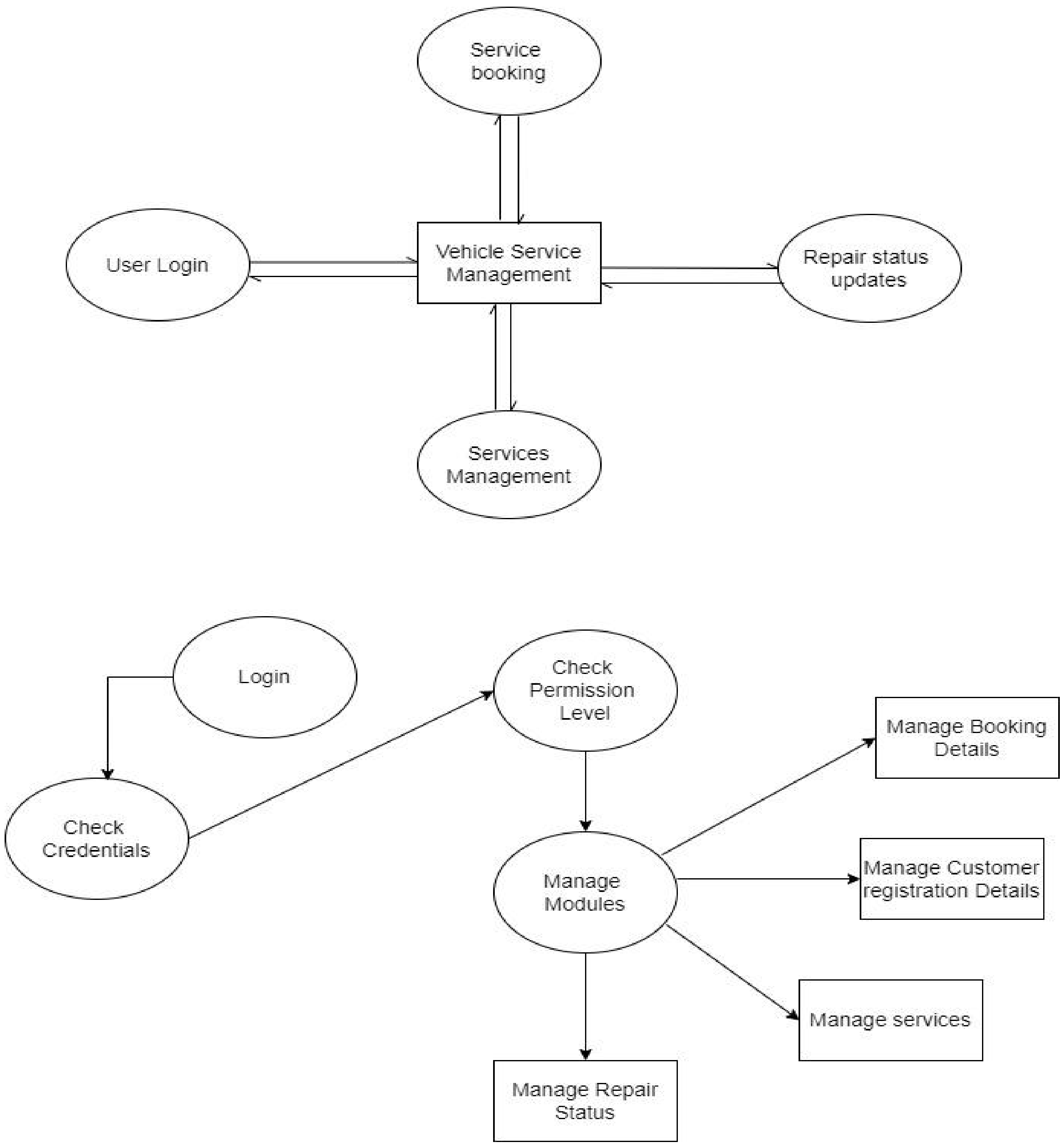


Figure 3.3 Data Flow Diagram

CHAPTER 4
SYSTEM IMPLEMENTATION

The implementation of various components of SteerX is provided in this chapter.

4.1 CODE IMPLEMENTATION

Python Django Framework is used as the technology stack to develop SteerX. Since, django is python based server code is saved with the .py extension. The web based HTML files are stored in the "templates" directory and its corresponding .css and .js files are linked using the django template language, from the "static" directory.

4.2 DEVELOPMENT ENVIRONMENT

The development environment is the bunch of processes and programming tools used to build the program or software. The following kits are used in the development of the proposed application.

- Python
- Pip
- Virtualenv Environment
- Visual Studio Code

4.3 APPLICATION USER INTERFACE

The Application User Interface refers to the features of the application which allows the user to interact with it. SteerX can be used to book vehicle services . SteerX can manage different user groups and their permission.

4.3.1 Login Page

The web application's home page is the login page shown in Figure 4.1. This page has a side navigation bar where end users (customers) can register, reset passwords, and access a home page reference which the admin user can configure.

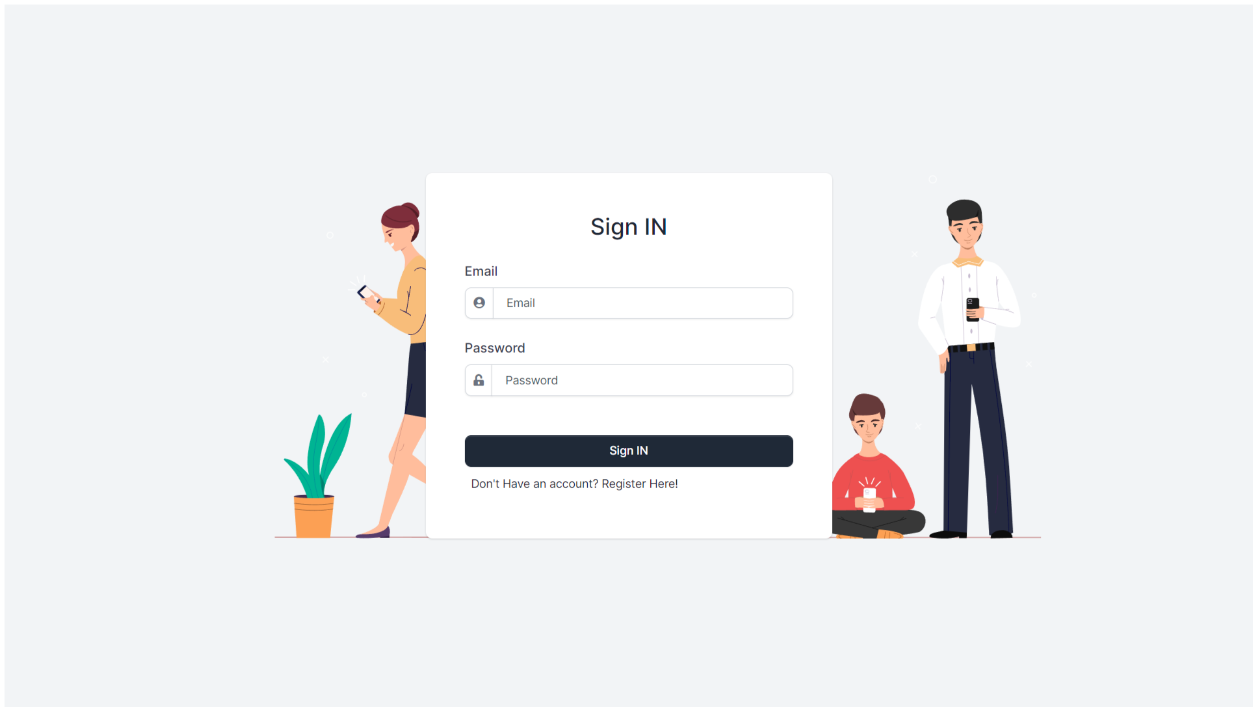


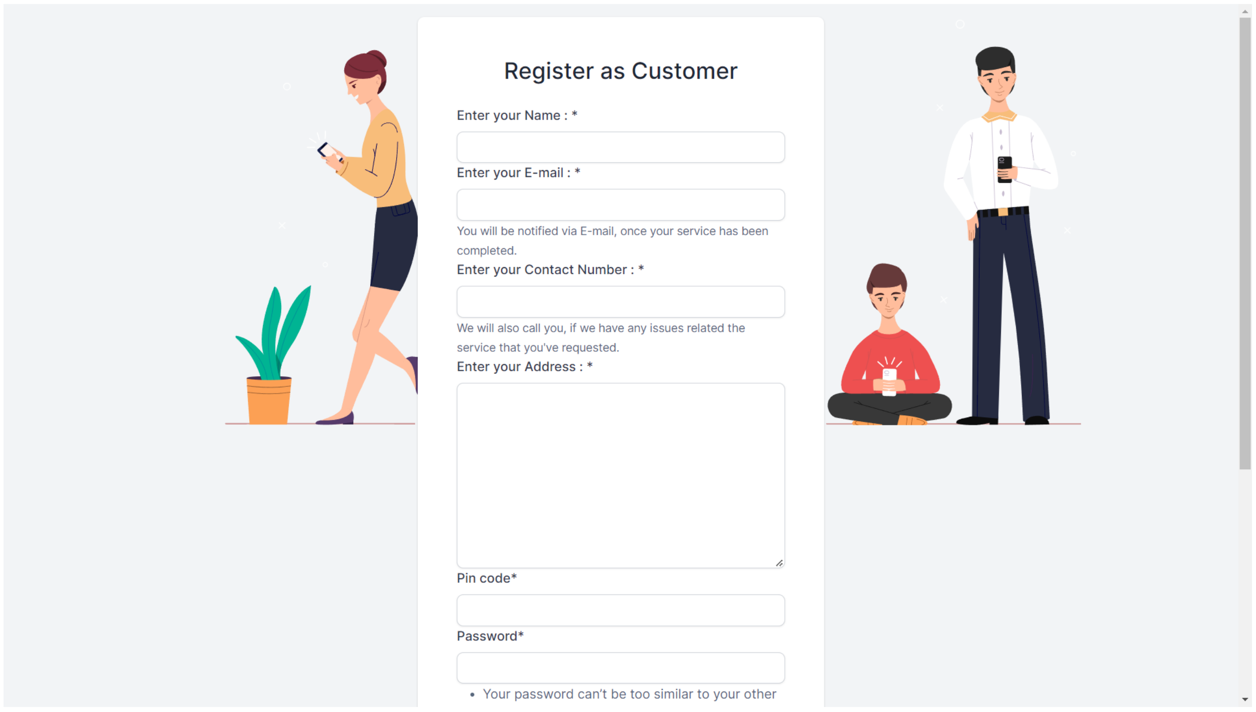
Figure 4.1 Login Page

4.3.2 Registration Page

The register page allows end users (customers) to register with their name, valid email address which will be checked later through account password, and a few other details as shown in Figure 4.2 and Figure 4.3. The recipient gets an email to confirm their account after sending the request.

CODE:

```
class NewCustomerForm(UserCreationForm):
    class Meta:
        model = get_user_model()
        fields = ['name', 'email', 'phone', 'address', 'pin_code']
        help_texts = {
        'email': "You will be notified via E-mail, once your service has 
been completed.",
        'phone': "We will also call you, if we have any issues related the 
service that you've requested."
        }

        labels = {
        'name' : ' Enter your Name : ',
        'email' : ' Enter your E-mail : ',
        'phone':'Enter your Contact Number : ',
        'address':'Enter your Address : ',
        'pin code':'Enter your pincode : ',
        }

    def save(self, commit=True):
        user = super(NewCustomerForm, self).save(commit=False)
        if commit:
            user.save()
        return user
```



Register as Customer

Enter your Name : *

Enter your E-mail : *

You will be notified via E-mail, once your service has been completed.

Enter your Contact Number : *

We will also call you, if we have any issues related the service that you've requested.

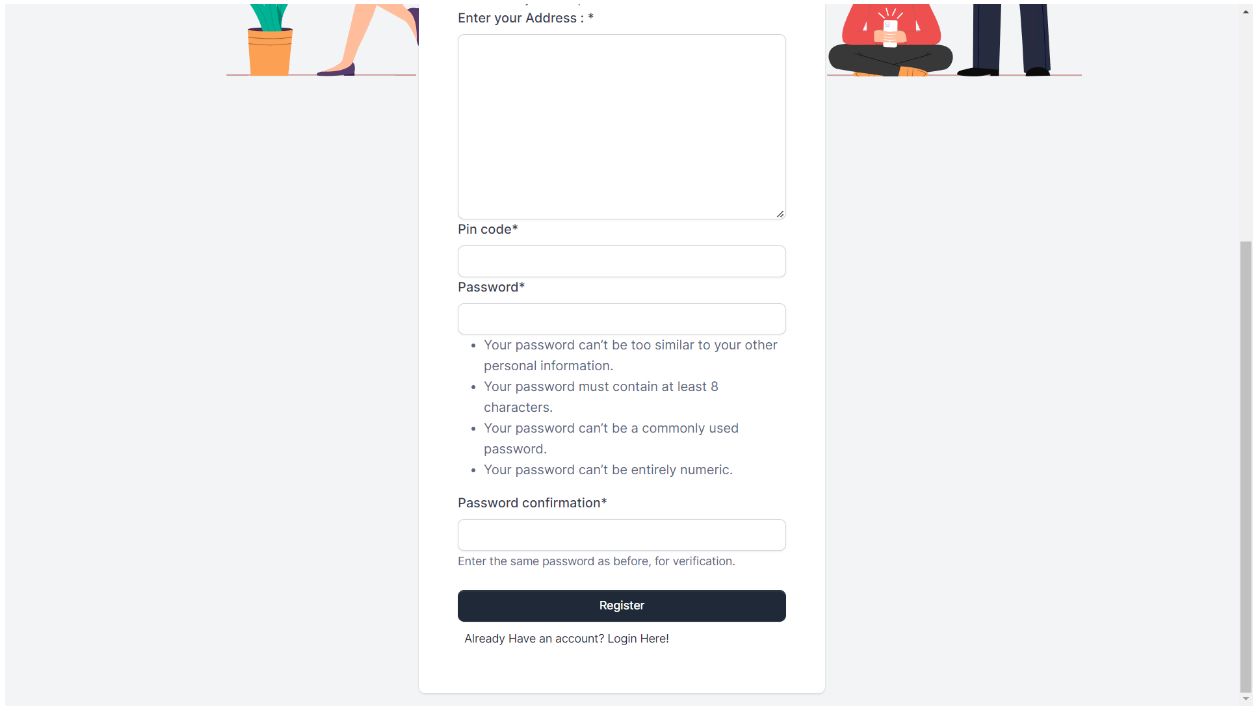
Enter your Address : *

Pin code*

Password*

- Your password can't be too similar to your other

Figure 4.2 Registration Page (1)


Enter your Address : *

Pin code*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Figure 4.3 Registration Page(2)

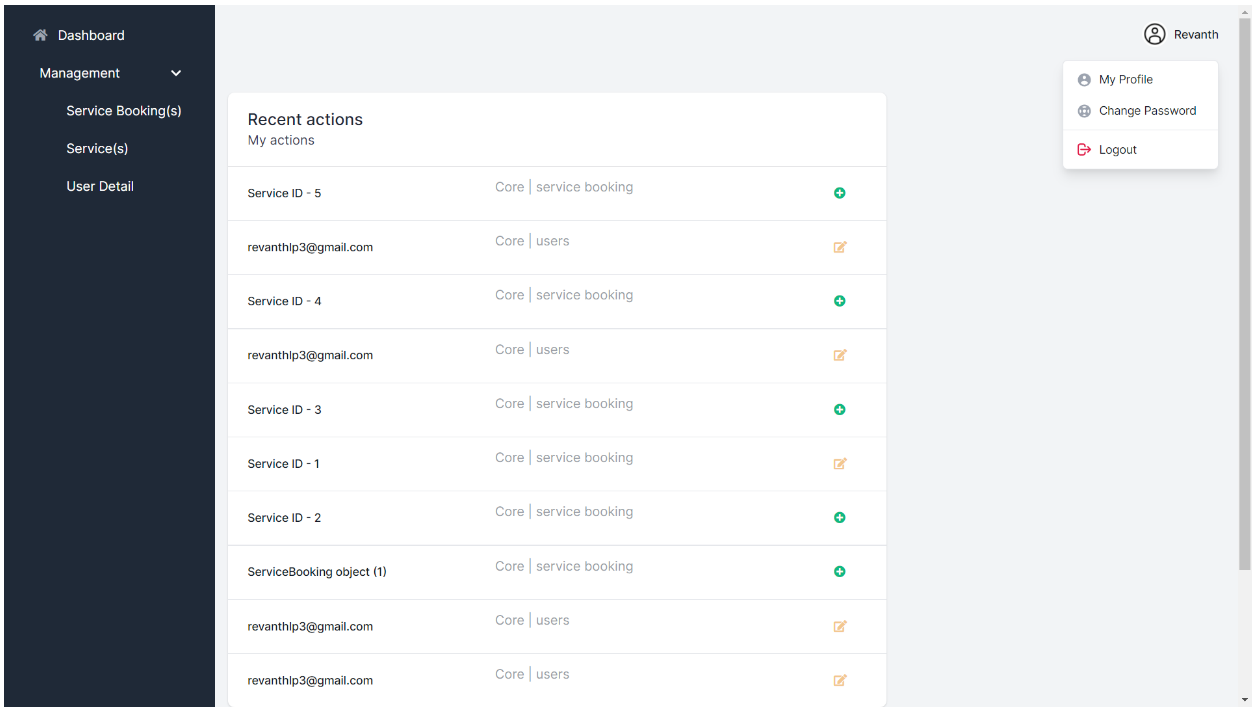
4.3.3 Customer

This allows customers to search for mechanics based on a variety of criteria such as vehicle type, mechanic availability, and so on. The consumer can book one or more services for their vehicle, as well as check the status of the services they have booked.This also allows them to edit their profile information.

CODE:

```
class CustomUserAdmin(UserAdmin):
    fieldsets = (
        (None, {'fields': ('email', 'password')}),
        (_('Personal info'), {'fields': ('name', 'phone', 'address',
'pin_code', 'mechanic_threshold')}),
        (_('Permissions'), {'fields': ('is_active', 'is_staff',
'is_superuser',
                                       'groups', 'user_permissions')}),
        (_('Important dates'), {'fields': ('last_login', 'date_joined')}),
    )
    staff_fieldsets = (
        (None, {'fields': ('email', 'password')}),
        (_('Personal info'), {'fields': ('name', 'phone', 'address',
'pin_code', 'mechanic_threshold')}),
        (_('Important dates'), {'fields': ('last_login', 'date_joined')}),
    )
    customer_fieldsets = (
        (None, {'fields': ('email', 'password')}),
        (_('Personal info'), {'fields': ('name', 'phone', 'address',
'pin_code')}),
        (_('Important dates'), {'fields': ('last_login', 'date_joined')}),
    )
    add_fieldsets = (
        (None, {
            'classes': ('wide',),
            'fields': ('email', 'name', 'phone', 'address', 'pin_code',
'password1', 'password2'),
        }),
    )
    list_display = ('email', 'name', 'phone', 'is_staff',)
    search_fields = ('name', 'phone', 'email', 'pin_code',)
    ordering = ('date_joined',)
    staff_readonly_fields = ('last_login', 'date_joined', 'email',)

    def get_readonly_fields(self, request, obj=None):
        if not request.user.is_superuser:
            return self.staff_readonly_fields
        else:
            return super(CustomUserAdmin,
self).get_readonly_fields(request, obj)

    def get_fieldsets(self, request, obj=None):
        if not request.user.is_superuser:
            if request.user.groups.filter(name='customers').exists():
                return self.customer_fieldsets
            return self.staff_fieldsets
        else:
            return super(CustomUserAdmin, self).get_fieldsets(request,
obj)
```

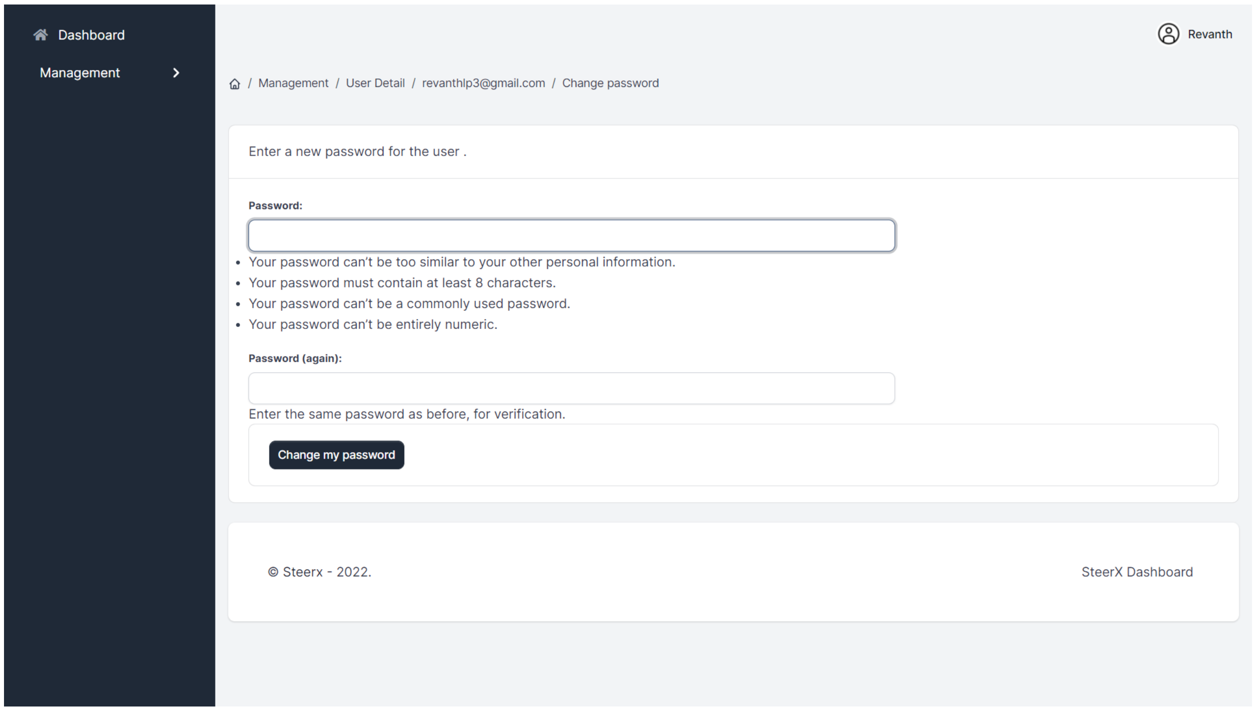
```python
def get_queryset(self, request):
    if not request.user.is_superuser:
        if request.user.groups.filter(name='customers').exists():
            return User.objects.filter(email=request.user.email)
    return User.objects.all()
```



Recent actions
My actions

Service ID - 5	Core | service booking
revanthlp3@gmail.com	Core | users
Service ID - 4	Core | service booking
revanthlp3@gmail.com	Core | users
Service ID - 3	Core | service booking
Service ID - 1	Core | service booking
Service ID - 2	Core | service booking
ServiceBooking object (1)	Core | service booking
revanthlp3@gmail.com	Core | users
revanthlp3@gmail.com	Core | users

Figure 4.4 Customer Dashboard


Dashboard
Management
Revanth
/ Management / User Detail / revanthlp3@gmail.com / Change password

Enter a new password for the user .

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password (again):

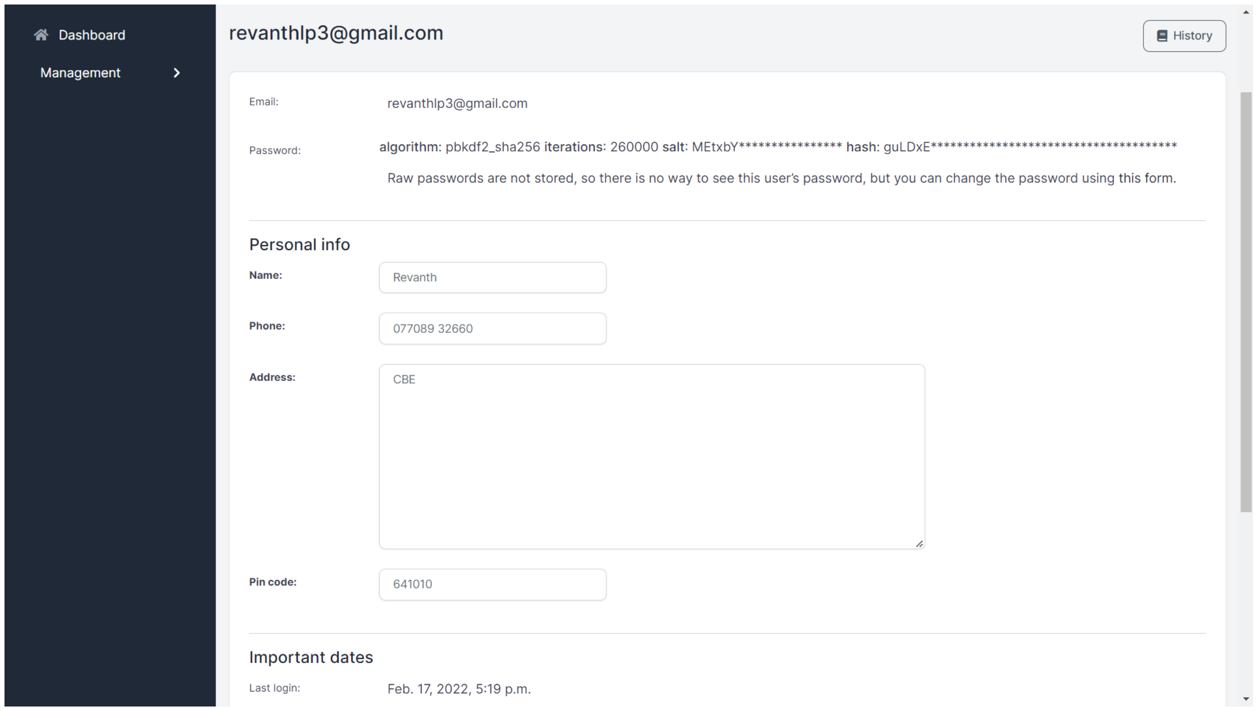
Enter the same password as before, for verification.

© Steerx - 2022.

SteerX Dashboard

Figure 4.5 Change Password

The Figure 4.4 depicts the customer dashboard on which the customer lands after logging in. The Figure 4.5 depicts the page where the user can change their password and create a new password. The Figure 4.6 depicts the user details page where the user may check and update their personal information and other details.



revanthlp3@gmail.com

Email:	revanthlp3@gmail.com
Password:	algorithm: pbkdf2_sha256 iterations: 260000 salt: MEtxbY*************** hash: guLDxE************************************
	Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

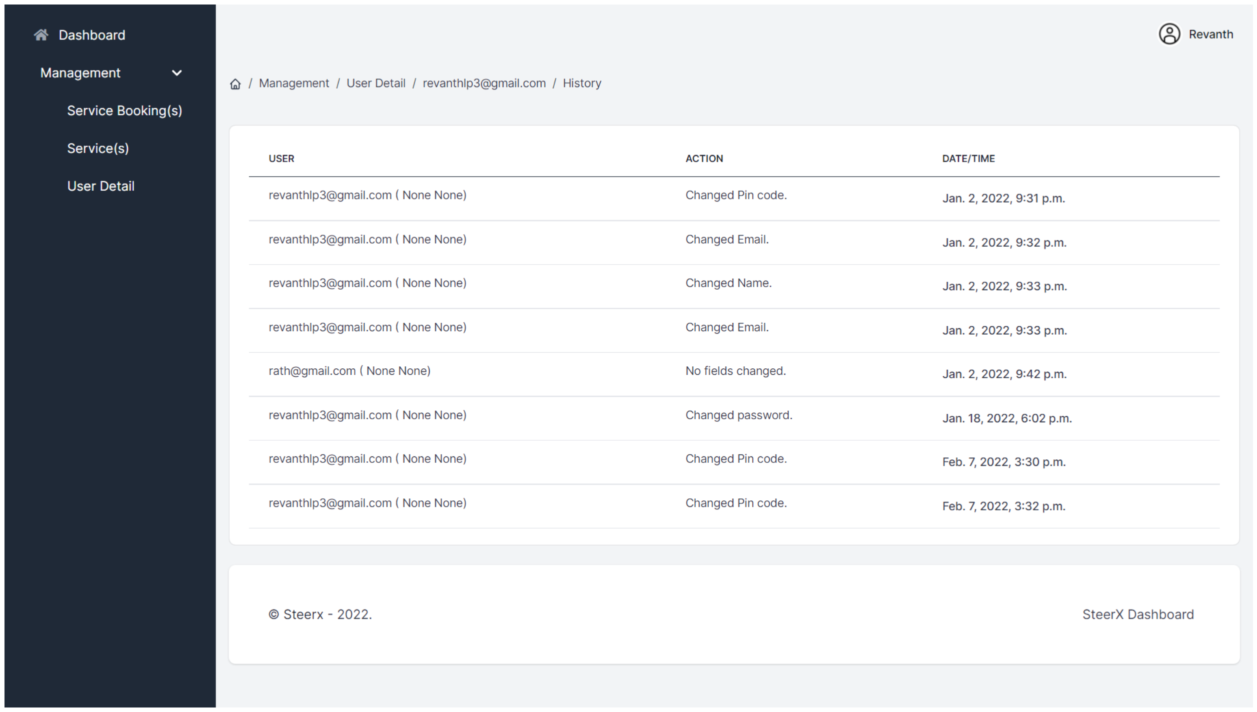
Personal info

Name:	Revanth
Phone:	077089 32660
Address:	CBE
Pin code:	641010

Important dates

Last login:	Feb. 17, 2022, 5:19 p.m.

Figure 4.6 User Profile

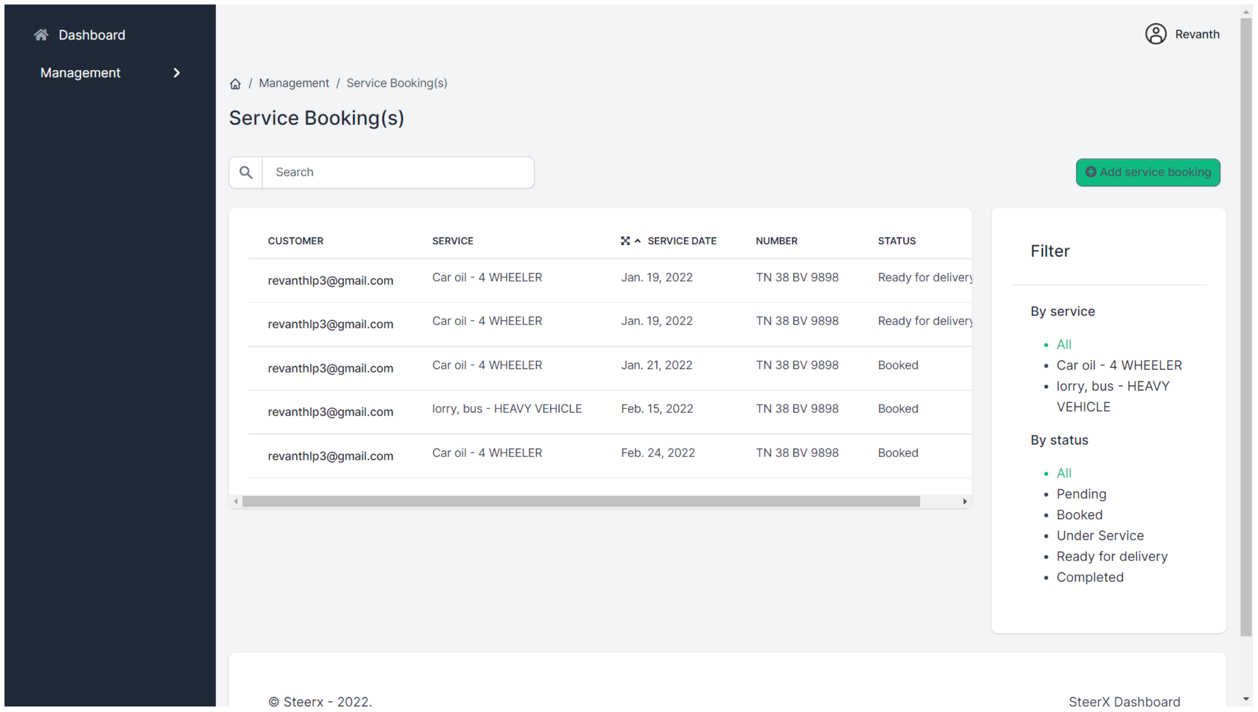


/ Management / User Detail / revanthlp3@gmail.com / History

USER	ACTION	DATE/TIME
revanthlp3@gmail.com (None None)	Changed Pin code.	Jan. 2, 2022, 9:31 p.m.
revanthlp3@gmail.com (None None)	Changed Email.	Jan. 2, 2022, 9:32 p.m.
revanthlp3@gmail.com (None None)	Changed Name.	Jan. 2, 2022, 9:33 p.m.
revanthlp3@gmail.com (None None)	Changed Email.	Jan. 2, 2022, 9:33 p.m.
rath@gmail.com (None None)	No fields changed.	Jan. 2, 2022, 9:42 p.m.
revanthlp3@gmail.com (None None)	Changed password.	Jan. 18, 2022, 6:02 p.m.
revanthlp3@gmail.com (None None)	Changed Pin code.	Feb. 7, 2022, 3:30 p.m.
revanthlp3@gmail.com (None None)	Changed Pin code.	Feb. 7, 2022, 3:32 p.m.

© Steerx - 2022. SteerX Dashboard

Figure 4.7 History



Service Booking(s)

CUSTOMER	SERVICE	SERVICE DATE	NUMBER	STATUS
revanthlp3@gmail.com	Car oil - 4 WHEELER	Jan. 19, 2022	TN 38 BV 9898	Ready for delivery
revanthlp3@gmail.com	Car oil - 4 WHEELER	Jan. 19, 2022	TN 38 BV 9898	Ready for delivery
revanthlp3@gmail.com	Car oil - 4 WHEELER	Jan. 21, 2022	TN 38 BV 9898	Booked
revanthlp3@gmail.com	lorry, bus - HEAVY VEHICLE	Feb. 15, 2022	TN 38 BV 9898	Booked
revanthlp3@gmail.com	Car oil - 4 WHEELER	Feb. 24, 2022	TN 38 BV 9898	Booked

Filter

By service

- All
- Car oil - 4 WHEELER
- lorry, bus - HEAVY VEHICLE

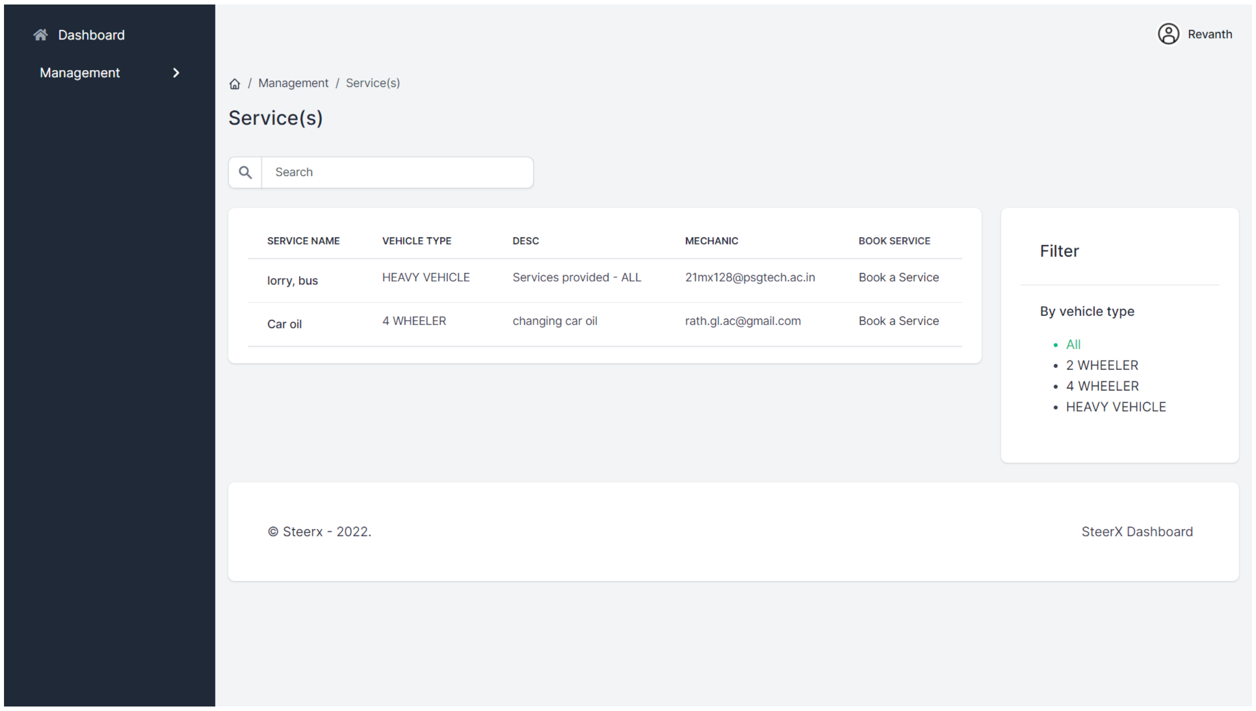
By status

- All
- Pending
- Booked
- Under Service
- Ready for delivery
- Completed

© Steerx - 2022. SteerX Dashboard

Figure 4.8 Service Bookings

Figure 4.7 depicts the history of changes made by the users. Figure 4.8 depicts the page where the user may check the services that they have previously booked. Figure 4.9 depicts the customer dashboard where the customer can search for one or more services based on their requirements. Figure 4.10 depicts the service booking page of the customer.



Service(s)

Search

SERVICE NAME	VEHICLE TYPE	DESC	MECHANIC	BOOK SERVICE
lorry, bus	HEAVY VEHICLE	Services provided - ALL	21mx128@psgtech.ac.in	Book a Service
Car oil	4 WHEELER	changing car oil	rath.gl.ac@gmail.com	Book a Service

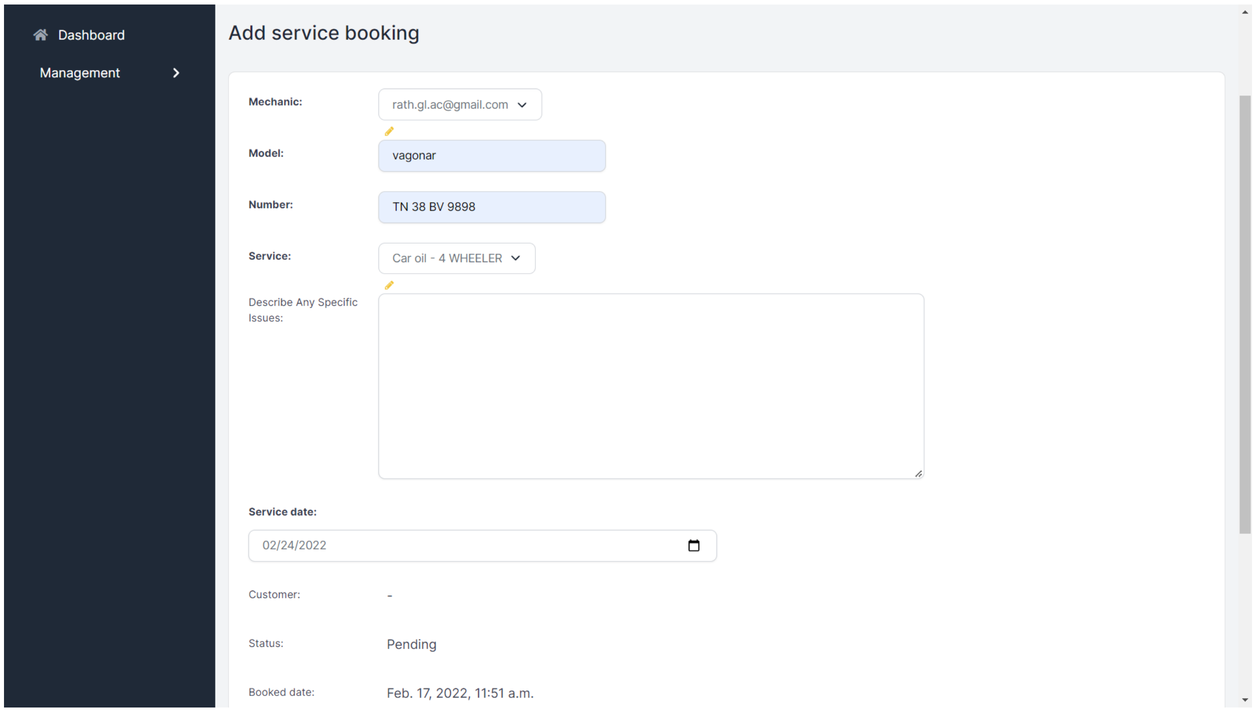
Filter

By vehicle type

- All
- 2 WHEELER
- 4 WHEELER
- HEAVY VEHICLE

© Steerx - 2022. SteerX Dashboard

Figure 4.9 Services



Add service booking

Mechanic:	rath.gl.ac@gmail.com
Model:	vagonar
Number:	TN 38 BV 9898
Service:	Car oil - 4 WHEELER
Describe Any Specific Issues:	
Service date:	02/24/2022
Customer:	-
Status:	Pending
Booked date:	Feb. 17, 2022, 11:51 a.m.

Figure 4.10 Service Booking

4.3.4 Mechanics

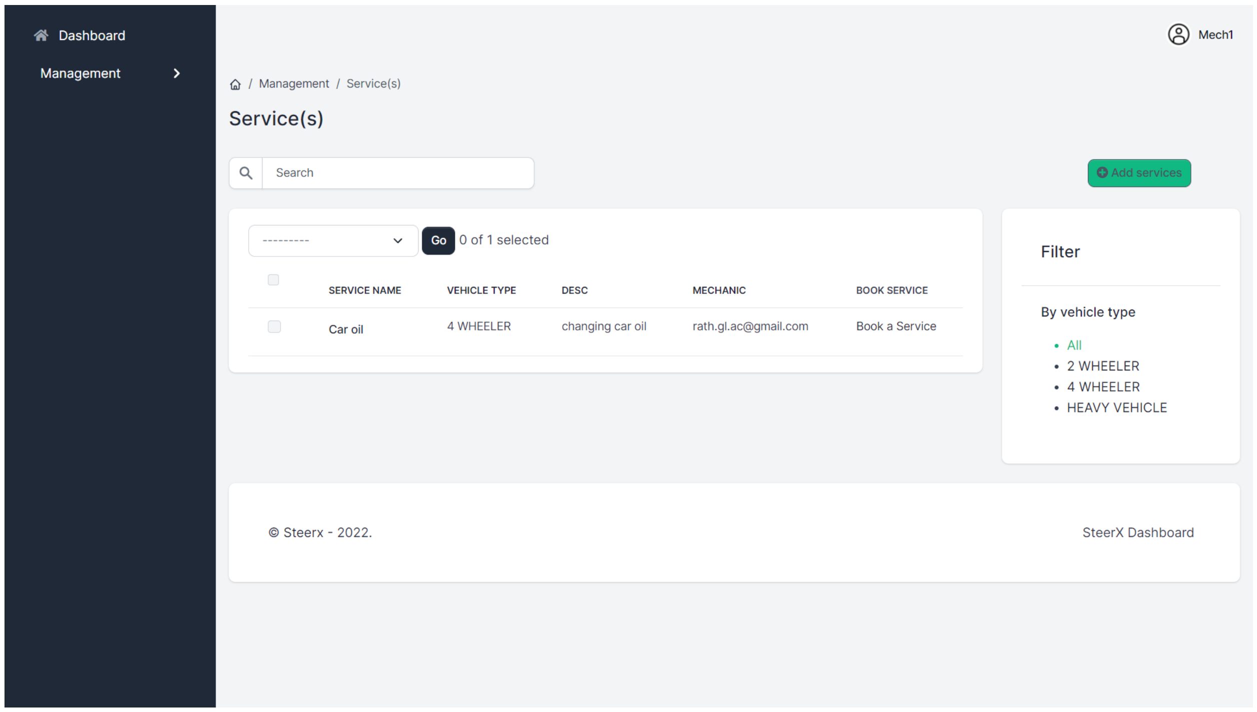
Mechanics can add one or more of the services which they provide. They may update the status of the booked vehicles and check the list of bookings for a certain date. When a customer requests a service, the mechanics will be notified by email. Every mechanic will have a threshold value that if the number of bookings exceeds the threshold value, the customer will be unable to book their services for that particular day.

CODE:

```
class ServicesAdmin(admin.ModelAdmin):
    list_display = ["service_name", "vehicle_type", "desc", "mechanic",
"getAddService"]
    search_fields = ["mechanic__email", "service_name", "desc"]
    list_filter = ["vehicle_type"]
    staff_readonly_fields = ('mechanic',)
    customer_readonly_fields = ('service_name', 'vehicle_type', 'desc',
'mechanic')

    def getAddService(self, ServicesAdmin):
        url = '/dashboard/core/servicebooking/add/'
        return mark_safe("""<a href="{url}"
target="_blank">{text}</a>""".format(url=url,text=_("Book a Service"),))
    getAddService.short_description = _("Book Service")

    def get_readonly_fields(self, request, obj=None):
        if not request.user.is_superuser:
            if request.user.groups.filter(name='customers').exists():
                return self.customer_readonly_fields
            return self.staff_readonly_fields
        else:
            return super(ServicesAdmin, self).get_readonly_fields(request,
obj)

    def get_queryset(self, request):
        if not request.user.is_superuser:
            if request.user.groups.filter(name='mechanics').exists():
                return
Services.objects.filter(mechanic__email=request.user.email)
```

```
            elif request.user.groups.filter(name='customers').exists():
                return Services.objects.all()
        return Services.objects.all()

    def save_model(self, request, obj, form, change):
        if not obj.pk and
request.user.groups.filter(name='mechanics').exists():
            obj.mechanic = request.user
        super().save_model(request, obj, form, change)
```

The Figure 4.11 depicts the mechanics dashboard where the mechanics can view a list of services provided by them. Figure 4.12 depicts the mechanic dashboard's add service page where the user can add one or more services provided by them.The Figure 4.13 depicts the status update page where mechanics update the status of vehicles scheduled for servicing.



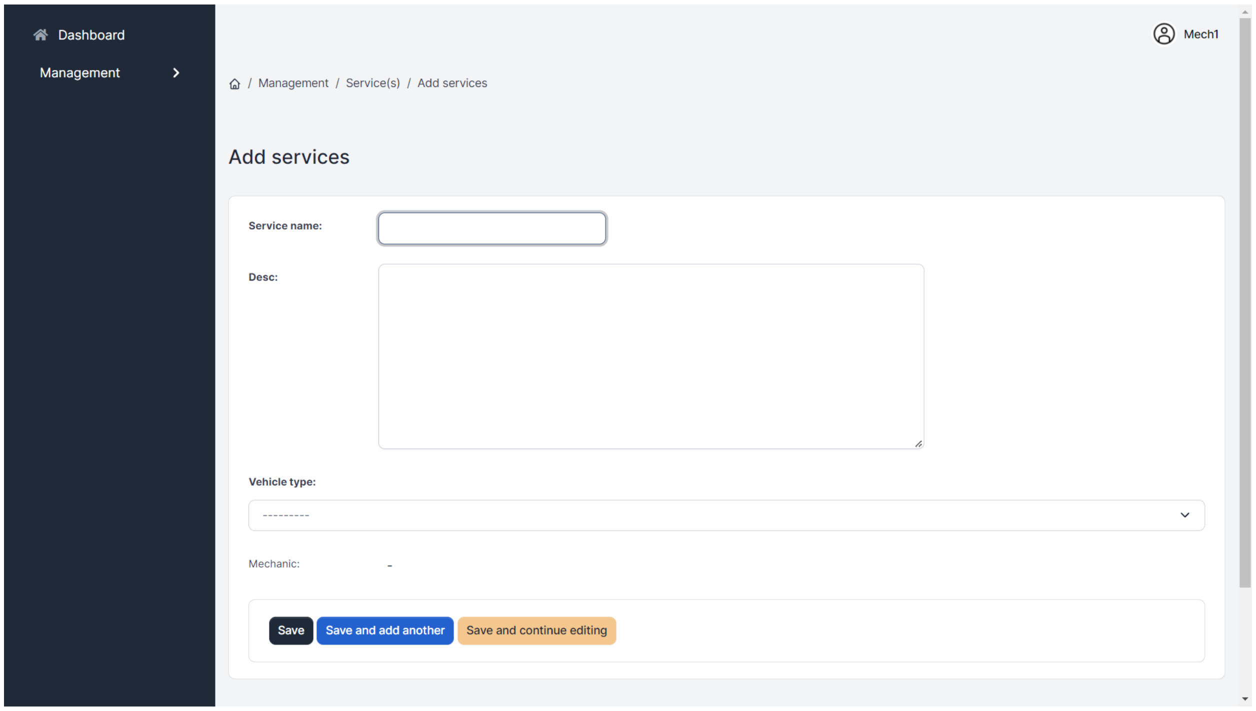
Service(s)

0 of 1 selected

SERVICE NAME	VEHICLE TYPE	DESC	MECHANIC	BOOK SERVICE
Car oil	4 WHEELER	changing car oil	rath.gl.ac@gmail.com	Book a Service

By vehicle type
- All
- 2 WHEELER
- 4 WHEELER
- HEAVY VEHICLE

Figure 4.11 Mechanics Dashboard


Dashboard
Management
Mech1
Management / Service(s) / Add services
Save
Save and add another
Save and continue editing

Add services

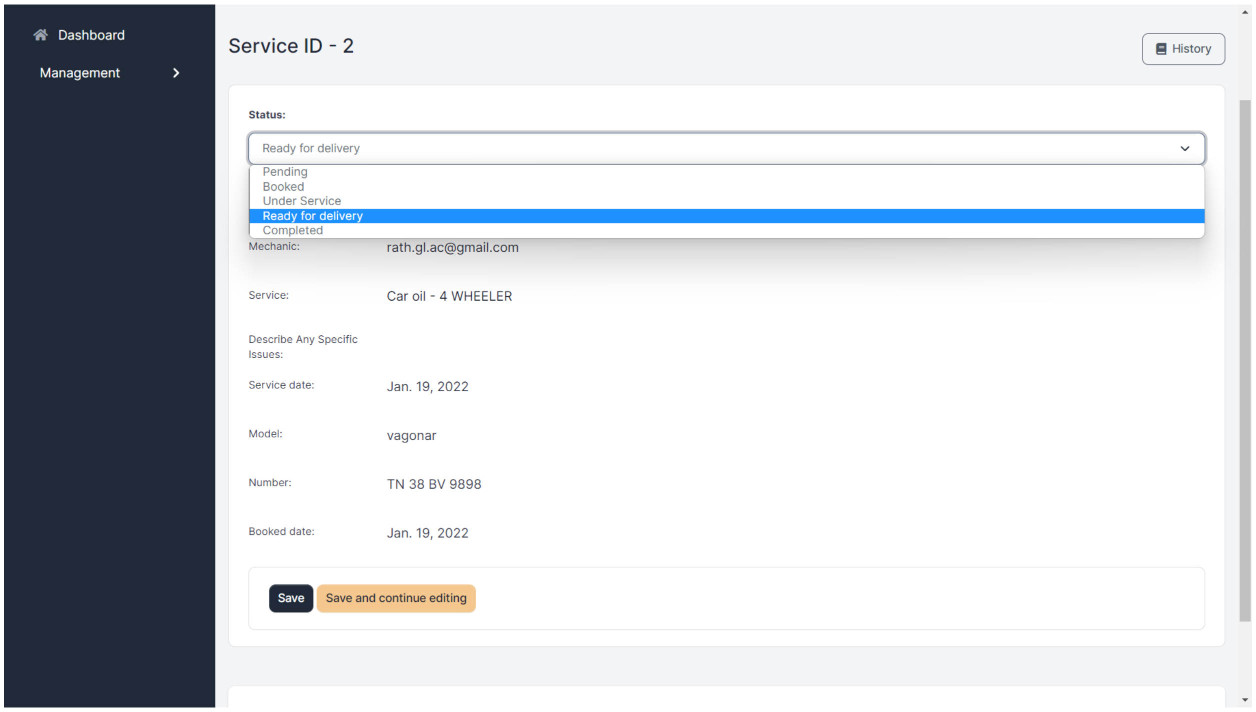
Service name:

Desc:

Vehicle type: ---------

Mechanic: -

Figure 4.12 Add Services



Service ID - 2

Status: Ready for delivery

Mechanic: rath.gl.ac@gmail.com

Service: Car oil - 4 WHEELER

Describe Any Specific Issues:

Service date: Jan. 19, 2022

Model: vagonar

Number: TN 38 BV 9898

Booked date: Jan. 19, 2022

Figure 4.13 Status Updation

4.3.5 Admin

Admin can add one or more users, modify all users permissions, verify booked services and their status, and check complaints about those services.

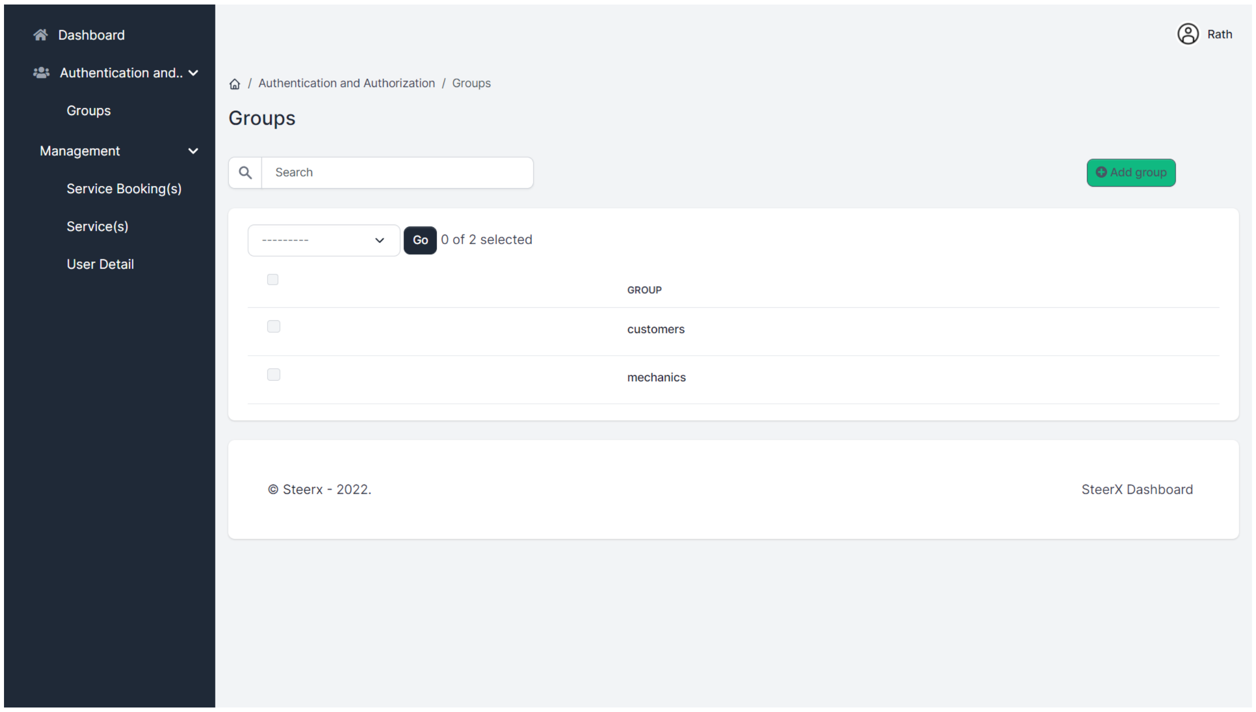
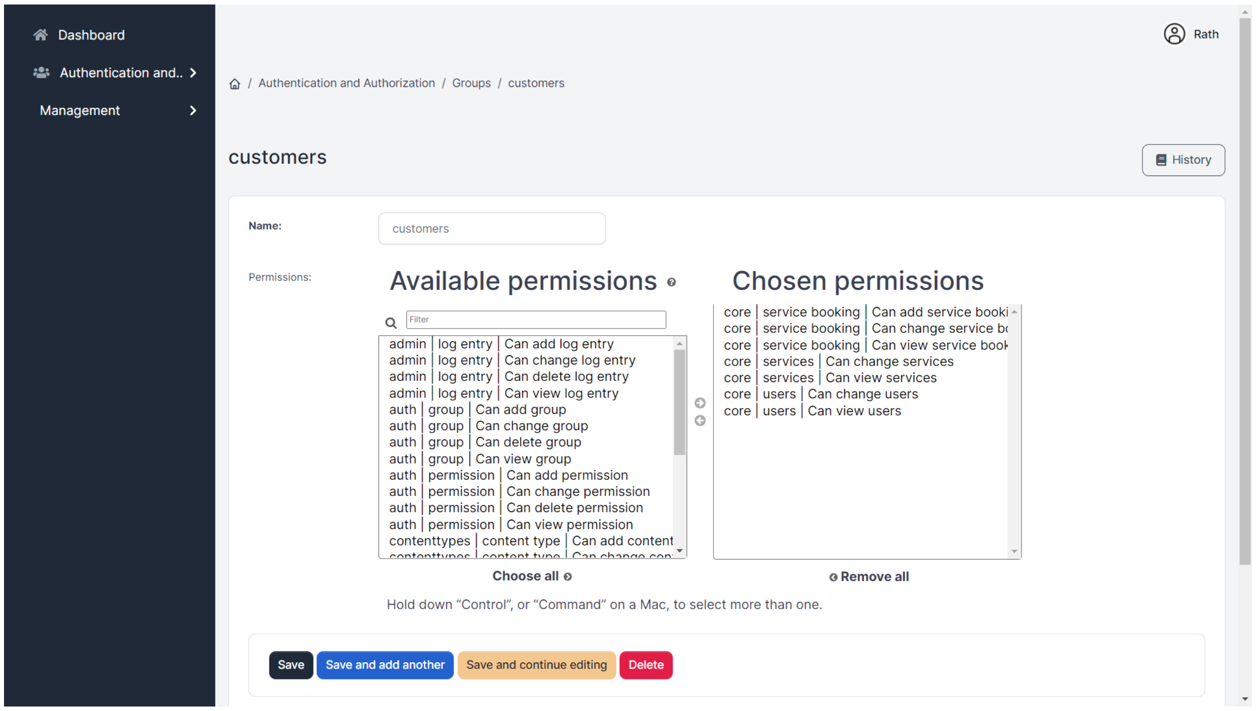


Figure 4.14 Admin Dashboard

Figure 4.14 depicts the admin dashboard. The Figure 4.15 depicts the permissions available for customers.The admin can review those permissions and can add or delete them.The Figure 4.16 depicts the permissions available for mechanics.The admin can review those permissions and can add or delete them.



Authentication and Authorization / Groups / customers

customers

Name: customers

Permissions:

Available permissions

Filter

admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry
auth | group | Can add group
auth | group | Can change group
auth | group | Can delete group
auth | group | Can view group
auth | permission | Can add permission
auth | permission | Can change permission
auth | permission | Can delete permission
auth | permission | Can view permission
contenttypes | content type | Can add content

Choose all

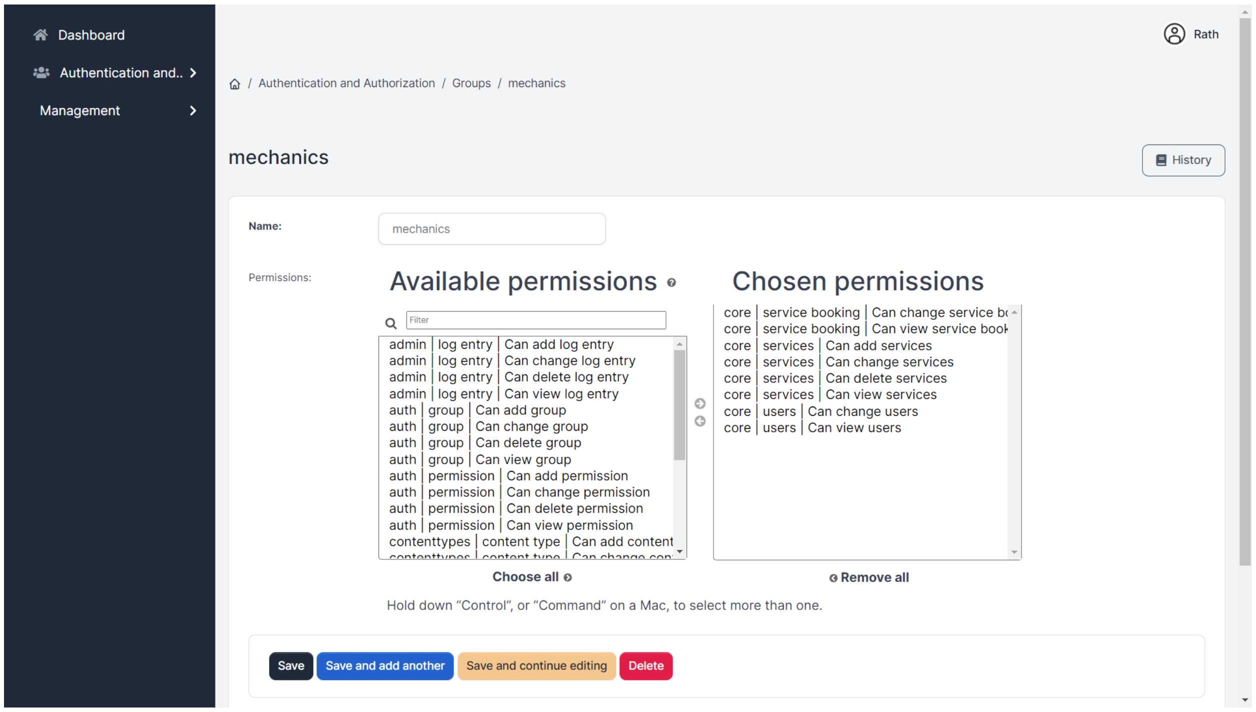
Chosen permissions

core | service booking | Can add service booki
core | service booking | Can change service bo
core | service booking | Can view service book
core | services | Can change services
core | services | Can view services
core | users | Can change users
core | users | Can view users

Remove all

Hold down "Control", or "Command" on a Mac, to select more than one.

Figure 4.15 Permissions(Customer)



Authentication and Authorization / Groups / mechanics

mechanics

Name: mechanics

Permissions:

Available permissions

Filter

admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry
auth | group | Can add group
auth | group | Can change group
auth | group | Can delete group
auth | group | Can view group
auth | permission | Can add permission
auth | permission | Can change permission
auth | permission | Can delete permission
auth | permission | Can view permission
contenttypes | content type | Can add content

Choose all

Chosen permissions

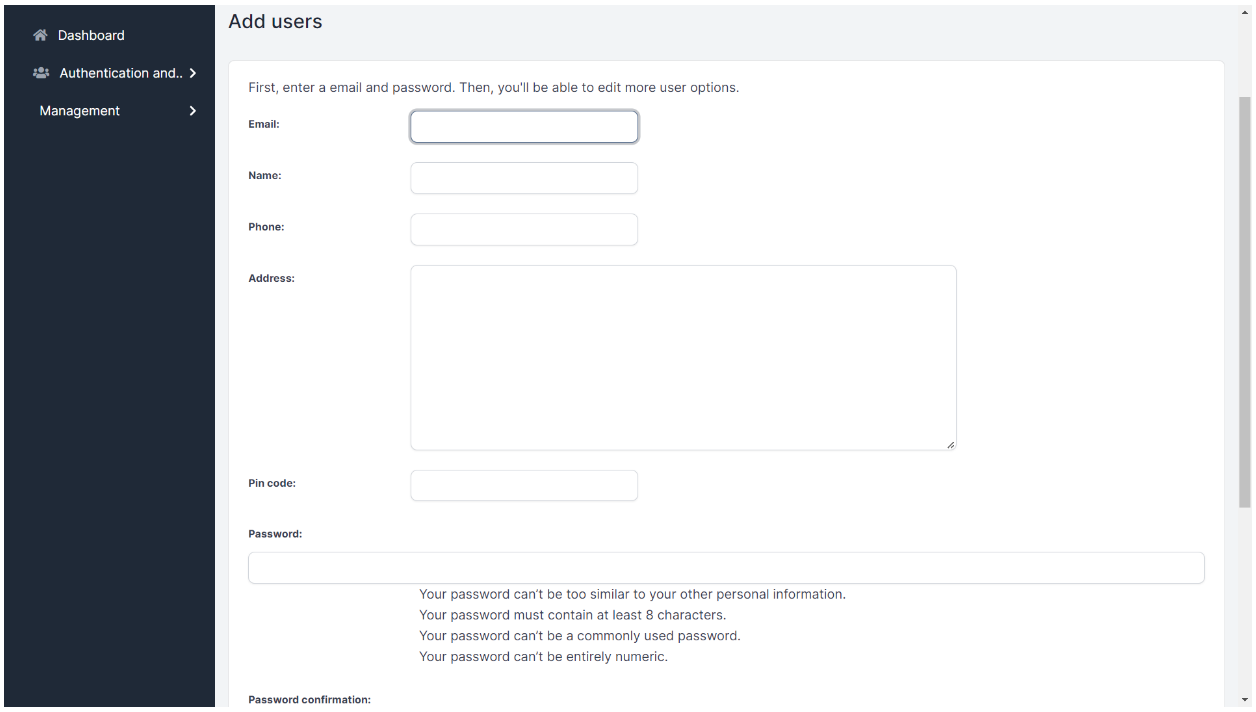
core | service booking | Can change service bo
core | service booking | Can view service book
core | services | Can add services
core | services | Can change services
core | services | Can delete services
core | services | Can view services
core | users | Can change users
core | users | Can view users

Remove all

Hold down "Control", or "Command" on a Mac, to select more than one.

Figure 4.16 Permissions(Mechanics)

Figure 4.17 and Figure 4.18 depicts the admin's add user page where the admin can add one or more mechanics.The Figure 4.19 depicts the user details page where the admin can view and update the personal information and other details of users.



Add users

First, enter a email and password. Then, you'll be able to edit more user options.

Email:

Name:

Phone:

Address:

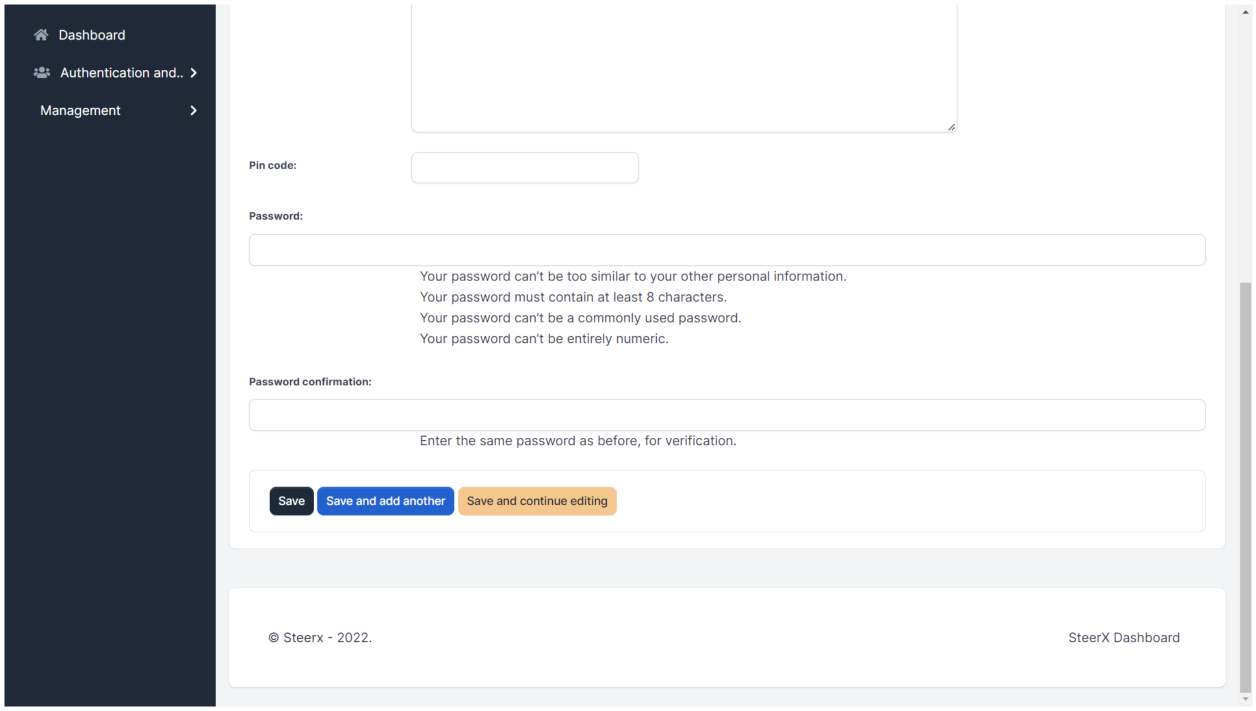
Pin code:

Password:

Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

Password confirmation:

Figure 4.17 Add Users (1)



Pin code:

Password:

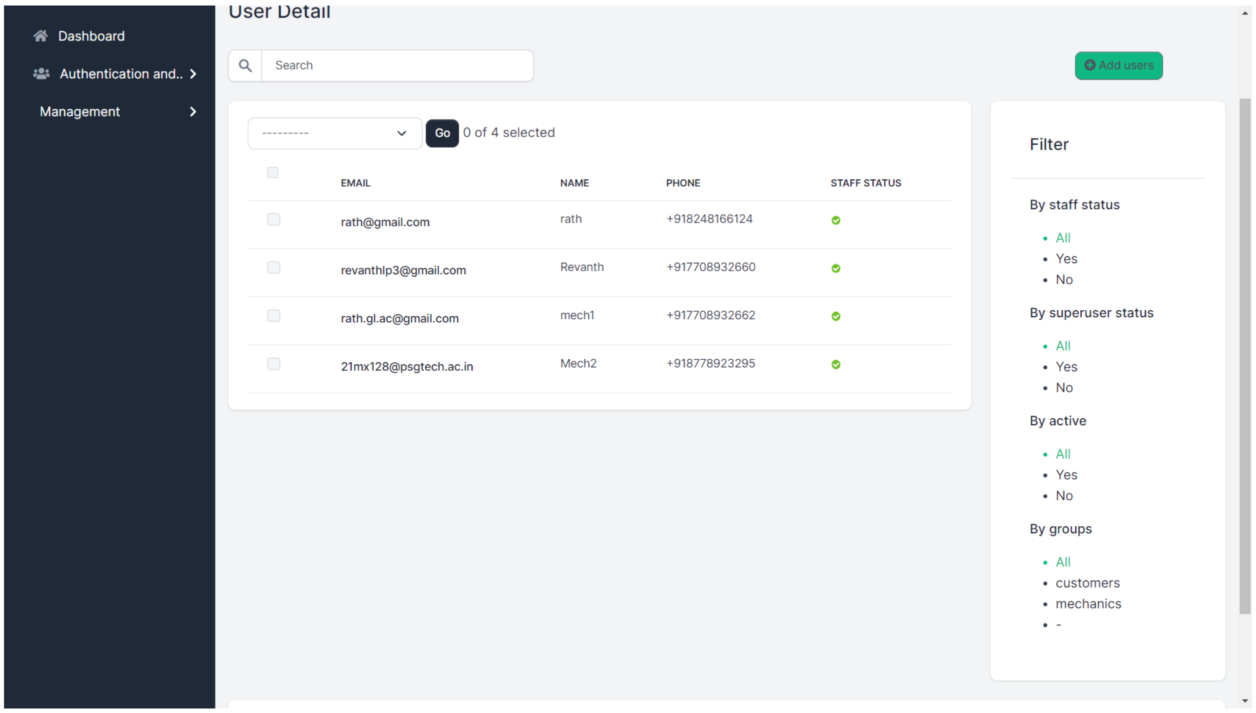
Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

© Steerx - 2022. SteerX Dashboard

Figure 4.18 Add Users (2)



0 of 4 selected

EMAIL	NAME	PHONE	STAFF STATUS
rath@gmail.com	rath	+918248166124	✓
revanthlp3@gmail.com	Revanth	+917708932660	✓
rath.gl.ac@gmail.com	mech1	+917708932662	✓
21mx128@psgtech.ac.in	Mech2	+918778923295	✓

Filter

By staff status
- All
- Yes
- No

By superuser status
- All
- Yes
- No

By active
- All
- Yes
- No

By groups
- All
- customers
- mechanics
- -

Figure 4.19 User Detail

CHAPTER 5

SYSTEM TESTING

In a software development cycle, software testing and quality assurance are extremely important. Both processes refine the entire process and ensure that the product is of superior quality. It also saves money on maintenance and provides better usability and functionality.

5.1 FUNCTIONAL TESTING

Functional testing is a quality assurance process and a type of black-box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output. Functional testing is conducted to evaluate the compliance of a system or component with specified functional requirements. Functional testing usually describes what the system does.This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation. Sample test cases for functional testing are given in Table 5.1.

Table 5.1 Test cases for Functional Testing

TEST ID	TEST CASE	EXPECTED OUTPUT	OBSERVED OUTPUT	RESULT
1	Login with valid email id and password	Dashboard of the user	Dashboard of the user	Success
2	Login with invalid email id and password	Show error to enter correct details	Show error to enter correct details	Success

5.2 USER INTERFACE TESTING

GUI Testing is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc. Sample test cases for functional testing are given in Table 5.2. The following are checklist to ensure the user interface testing,

- Check all the GUI elements for size, position, width, length, and acceptance of characters or numbers
- Check the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check Font used in an application is readable
- Check the alignment of the text is proper
- Check the Color of the font and warning messages is aesthetically pleasing
- Check that the images are properly aligned

Table 5.2 Test cases for User Interface Testing

TEST ID	TEST CASE	EXPECTED OUTPUT	OBSERVED OUTPUT	RESULT
1	Login with valid email id and password	Dashboard of the user	Dashboard of the user	Success
2	Login with invalid email id and password	Show error to enter correct details	Show error to enter correct details	Success

CHAPTER 6

CONCLUSION

SteerX has all of the essential features for booking services. It has a user interface that allows all groups of users to easily navigate and interact with the website. SteerX has been successfully implemented, thanks to the advancement of technology. SteerX would undoubtedly minimize human effort and simplify the job of customers who seek vehicle services and mechanics who provide them. Customers can, however, choose mechanics based on their favorite category, and since it is a web application, a mobile app can be developed as well. In that case, the proposal will be enhanced in the future to provide the solution.

BIBLIOGRAPHY

BOOKS

[1] Eric Matthes, "Python Crash Course, 2nd Edition", No Starch Press, 2019.

[2] Samuel Dauzon, "Django Essentials", Packt Publishing, 2014.

REFERENCES

[1] https://django.readthedocs.io/en/latest/

[2] https://docs.djangoproject.com/en/3.1/ref/contrib/admin/

[3] https://readthedocs.org/projects/django/

[4] https://docs.djangoproject.com/en/3.1/