ParkingPlus

1 INTRODUCTION

ParkingPlus is the application to manage multi floor parking system. Now a days search for a parking in metro cities is very time consuming task. This application tells admin about the available parking in building which could be further displayed on board. It stores the information about the total and available parking on each floor and entry/exit information of each car.
This application implements paid parking system. The charges are as below-

- $3 per hour till first 6 hours
- If parked for more than 6 hour, then $20 for per day parking.
- If parked for more than one day, then $20 per day, even if it exceeds a minute.

ParkingPlus application provide option to generate reports in csv excel format to see history of car parked in parking in some date range.

2 REQUIREMENTS

This application is using below python version and modules-

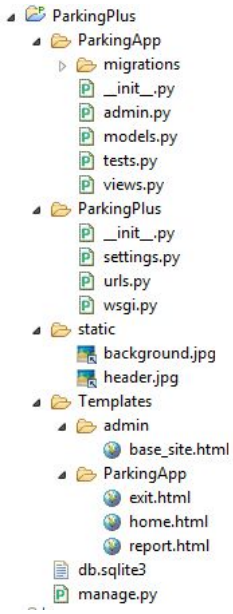
- Python version 2.7 and above
- Module Django 1.7
- Module datetime
- Database Sqlite3

To **run** the application follow below steps –

1. Go to command prompt
2. Direct it to the path where you saved the ParkingPlus folder as D:/PythonCourse/ParkingPlus
3. Run command 'python manage.py runserver'
4. Open the browser
5. Type localhost:8000 or 127.0.0.1:8000, It will show you home page

3 DESCRIPTION OF PYTHON PROGRAM

ParkingPlus is developed using Django module. Django provide its own MVC architecture to develop web application. Below screen shot shows the directory structure of development.



Directory structure of application

1. ParkingPlus directory is main directory has all the folders as shown in above picture. It has db.Sqlite3 to store database and manage.py to run the application
2. Sub ParkingPlus directory has __init__.py, settings.py, url.py and wsgi.py.
    a. settings.py has all the setting to run the application and directory paths for template and static files.
    b. url.py has the get/post urls for different webpages. By default the application start on home page.

```
from django.conf.urls import patterns, include, url
from django.contrib import admin

urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'ParkingPlus.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    url(r'^admin/', include(admin.site.urls)),
    url(r'^$', 'ParkingApp.views.home', name='home'),
    url(r'^carentry/', 'ParkingApp.views.carentry', name='carentry'),
    url(r'^exit/', 'ParkingApp.views.carsearch', name='exit'),
    url(r'^carexit/', 'ParkingApp.views.carexit', name='carexit'),
    url(r'^report/', 'ParkingApp.views.report', name='report'),
    url(r'^getreport/', 'ParkingApp.views.getreport', name='getreport'),
)
```

url.py

3. ParkingApp directory has __init__.py, models.py, views.py, admin.py python code files.
    a. This application is using two database tables, one to store the floor parking information and other to store car entry/exit and parking information. Models.py has two models for each table. Below screen shot shows the definition of each model.

```python
from django.db import models

# Create your models here.
#model for floor plan
class FloorPlan(models.Model):
    floorNo = models.SmallIntegerField(unique = True)
    totalParking = models.IntegerField()
    availableParking = models.IntegerField(null = True)

    def __str__(self):
        return str(self.floorNo)

#model for car entry exit information
class CarEntryExit(models.Model):
    carNo = models.CharField(max_length= 100)
    floorNo = models.SmallIntegerField()
    timeEntered = models.DateTimeField()
    timeExit = models.DateTimeField(null = True)
    feePaid = models.FloatField(null = True)

    def __unicode__(self):
        return self.carNo
```

models.py

b. admin.py has the python code to implement default admin functionality to manage the application. I added the list of model variable to show them in admin page grid and list of filter to apply the filters on grid to filter the data.

c. View.py is the main page for the application which implements different classes and methods to get and post request from the user. It has below classes to create the forms

   i. CarEntryForm() – class to create the form section for home page. It has 3 input fields-
      1. carNo. – required CharField with max length 10
      2. floorNo- required IntegerField, accept only integer value
      3. timeEntered – Read only DateTimeField field, initialized with current date time.
      4. It has custom validation for floorNo, which except the existing floor numbers and that floor must have available parking.

   ii. CarSearchForm() – class to create car search form on exit page. It has two input fields-
      1. carNo. – required CharField
      2. timeExit – Read only DateTimeField field, initialized with current date time.
      3. It has custom validation for carNo, the input carNo must exist in the database.

   iii. ReportForm() – class to create form section on report page. It has two input fields-
      1. fromDate – required DateField, accept date only in yyyy-MM-dd format.
      2. toDate – required DateField, accept date only in yyyy-MM-dd format.

This view.py has the function definitions to render get/post requests. Below is the small description of each methods defined in views.

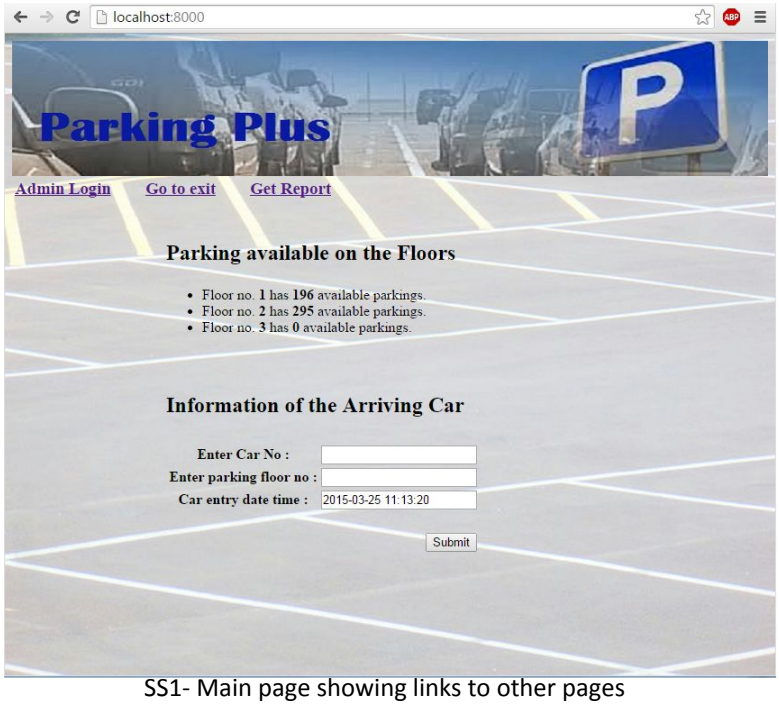
   iv. home() – renders the home page. It also passes the list of available parking on each floor to display data on home page.

v. Carentry() – invokes on submit POST on home page. It checks the validity of input data then save the car entry information in database and deduct the available parking of input floor.
vi. carsearch() – renders the exit page and initialize the current timeExit on each request.
vii. carexit() – invokes on submit request on exit page. It checks the validity of input data then check if car has already been exited. If not the save the exit time to DB and then calculate the parking fee and displays it on exit page. It add the free parking on that particular floor from where the current car exited.
viii. calculateFee() – takes the time difference of car entry and exit time in seconds as input, then apply the logic to calculate the fee as shown in introduction part.
ix. report() – renders the report page.
x. getreport() – invokes on getreport request from report page and generates the report in csv excel format. It includes all cars which were parked between from and to date means those cars which were entered after fromDate and exited before toDate.

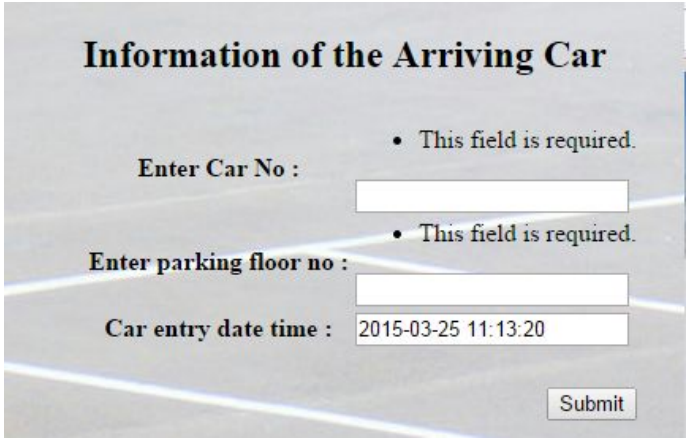
4. Static directory is to store static files. It has two images for page header and background.
5. Template directory has all the html template pages. Admin/base_site.html is to customize the admin view pages header. Html pages in ParkingApp folder has the template for home, exit and report pages.

4 SCREEN SHOT OF PROGRAM OUTPUT

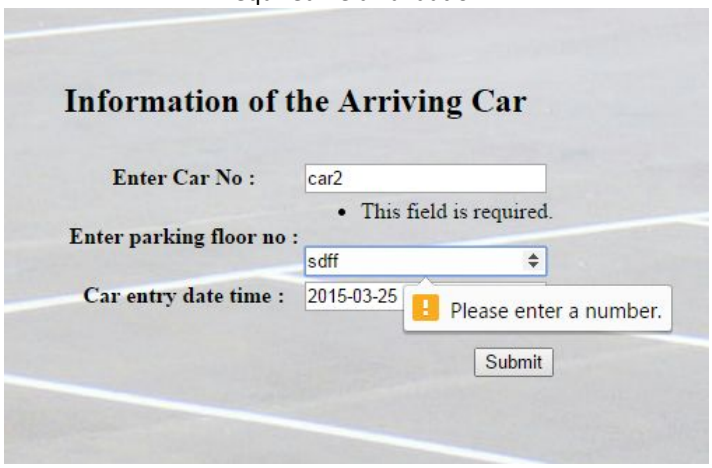
1- Home Page screens – localhost:8000



SS1- Main page showing links to other pages



Required field validation

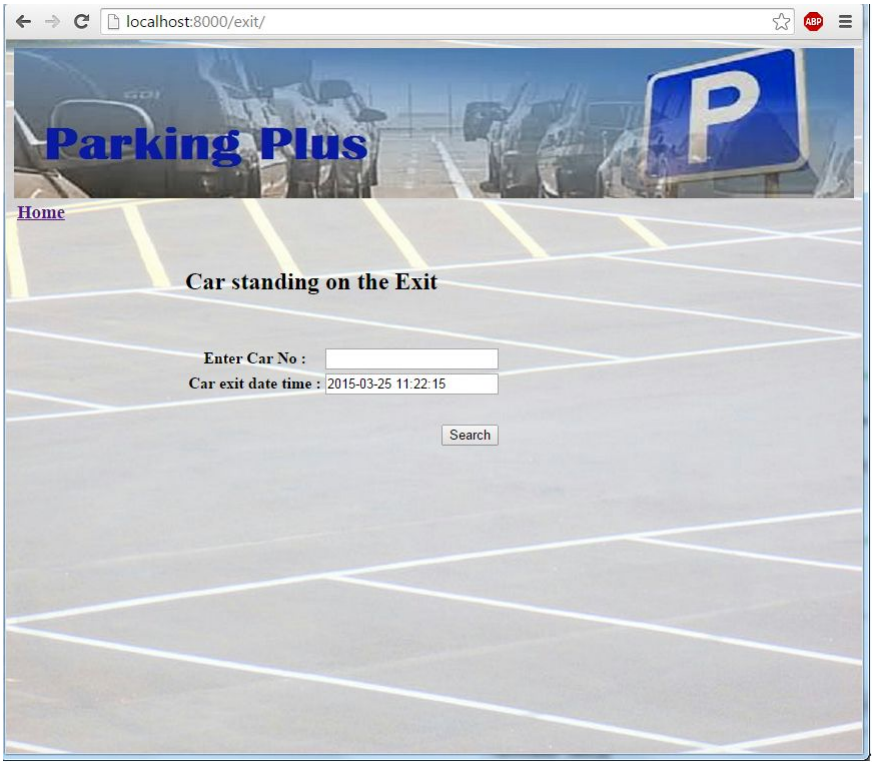


Allows only integer floor number

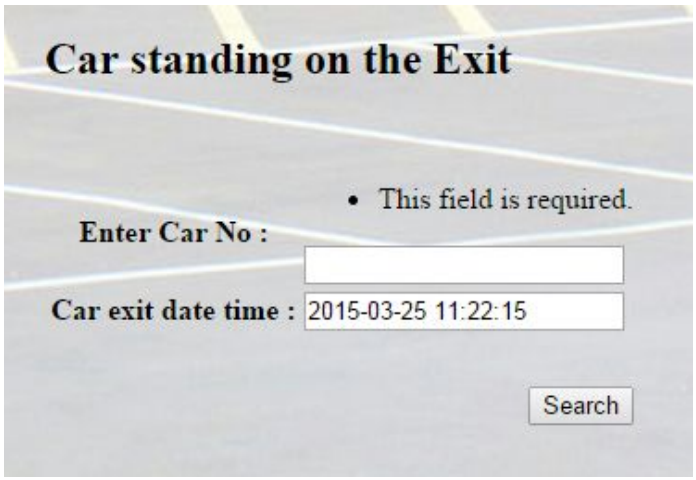
Shows error is floor no does not exist

Shows error is no available parking on input floor

2- Exit Page screen shots – Click on **<u>Go to Exit</u>** link on main page

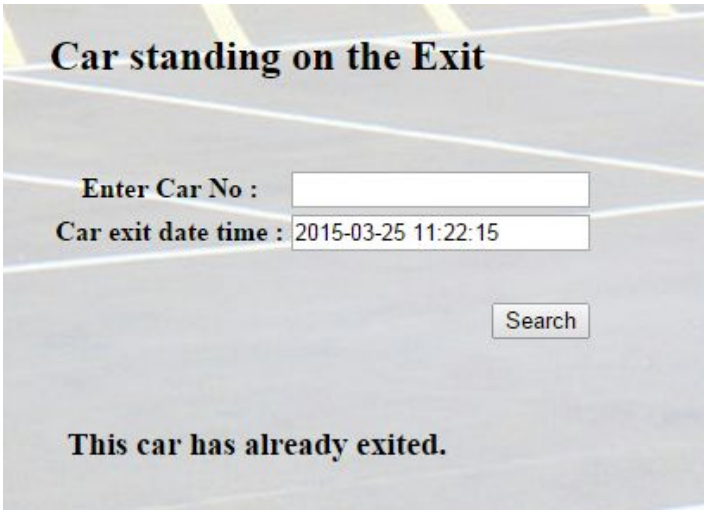


Exit page



Required field validation

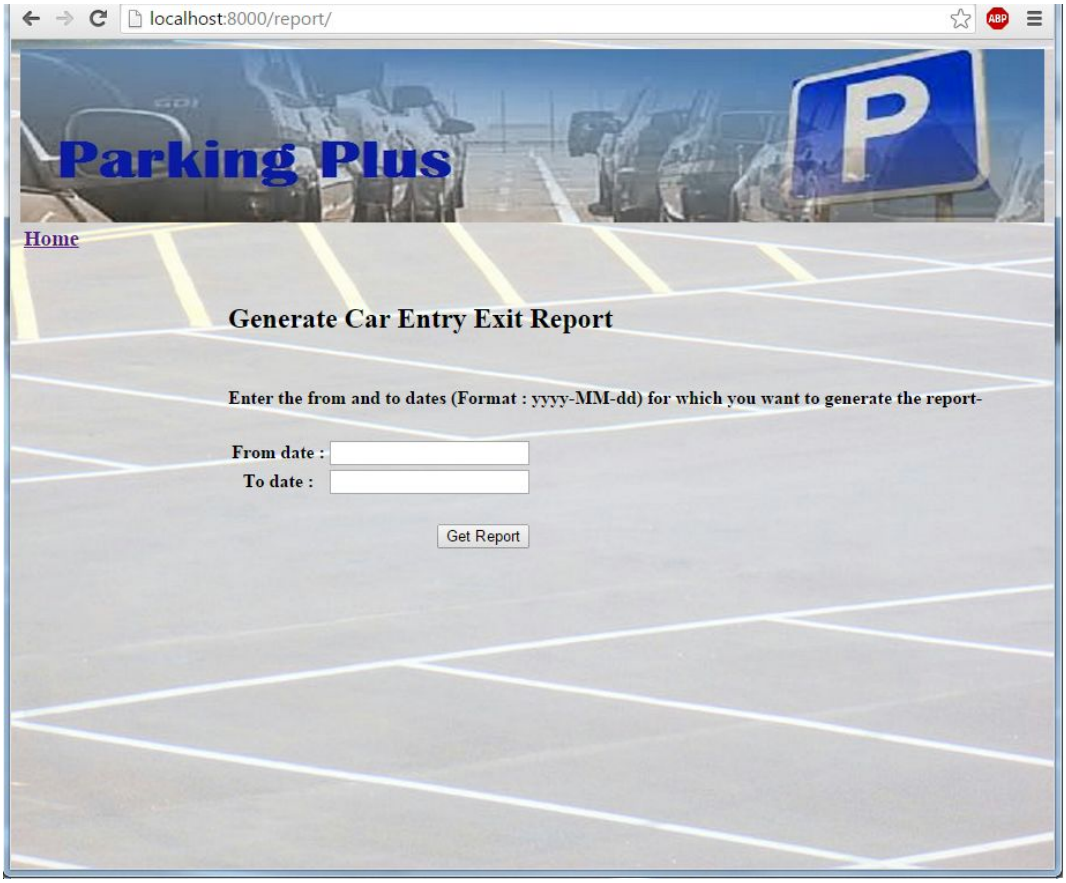
Shows error is car does not exist



Shows message if searched car has already exited

Shows the entry time of searched car and the parking fee

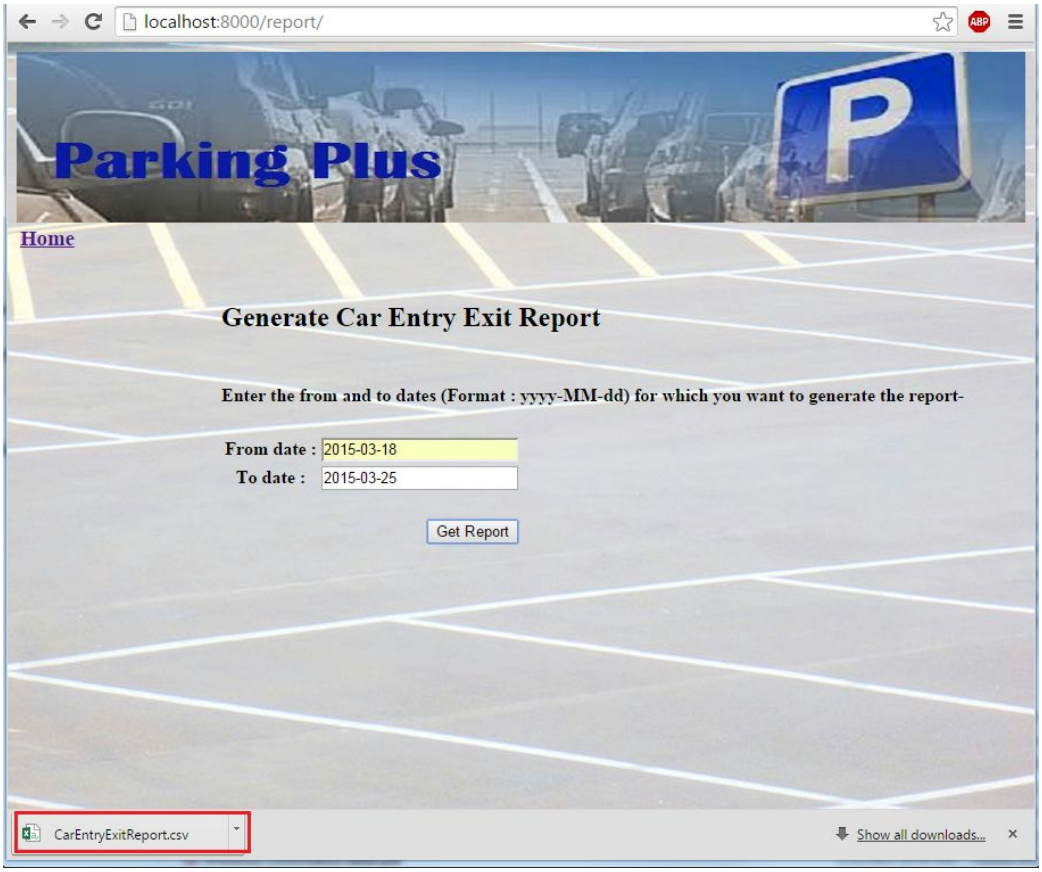
3- Report Page – Click on **<u>Get Report</u>** link on home page.



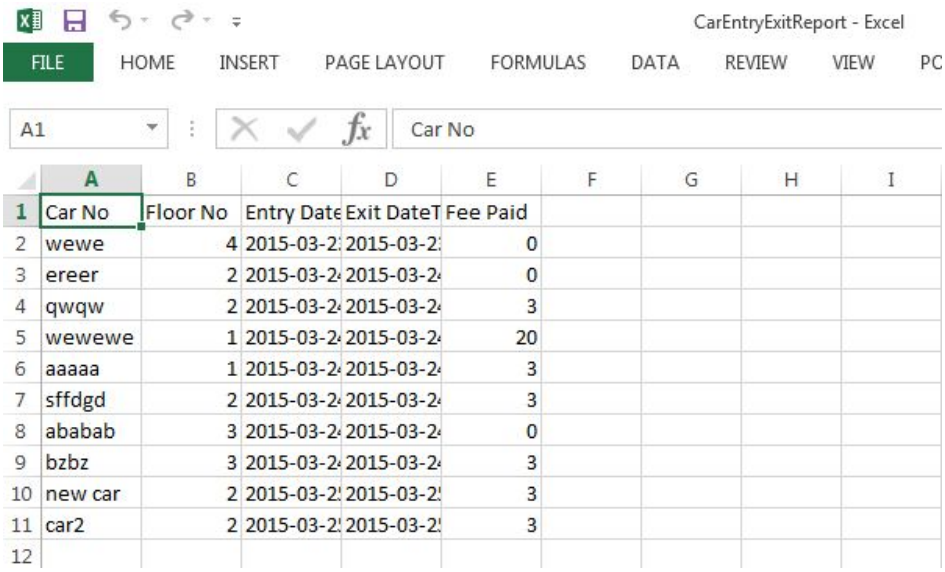
Report Page

Required field validation

Shows messge is the date is not in valid format



Download the report in csv formatted file



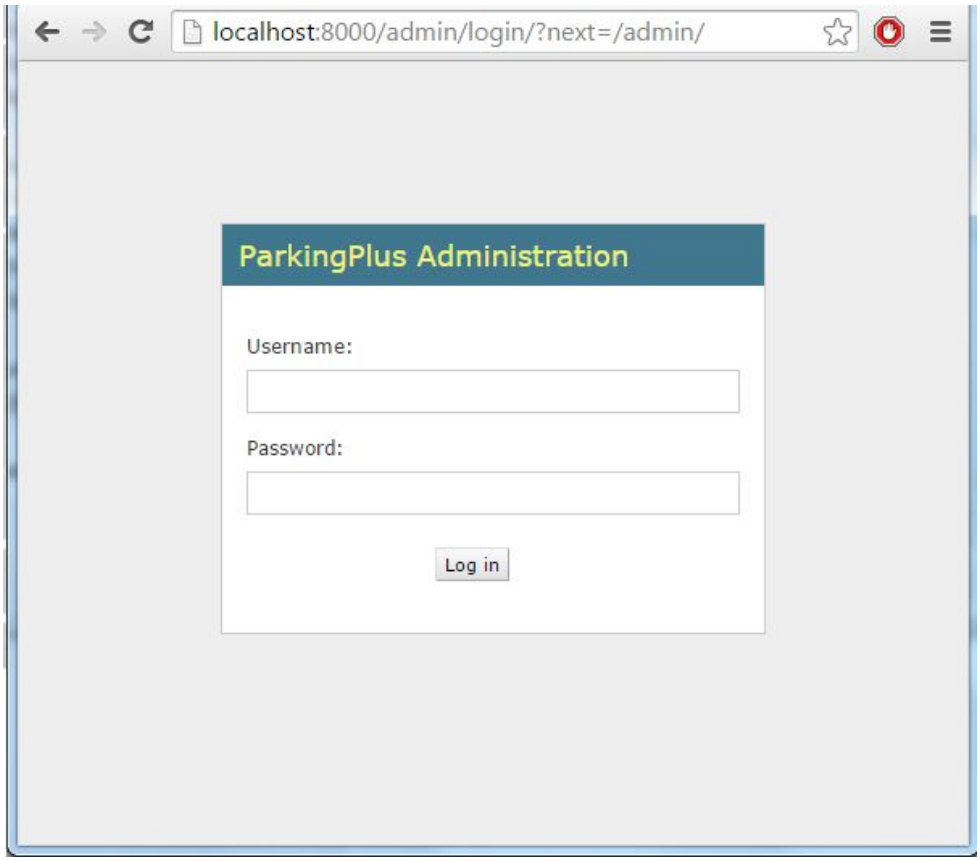
CarEntryExitReport - Excel

A1 | Car No

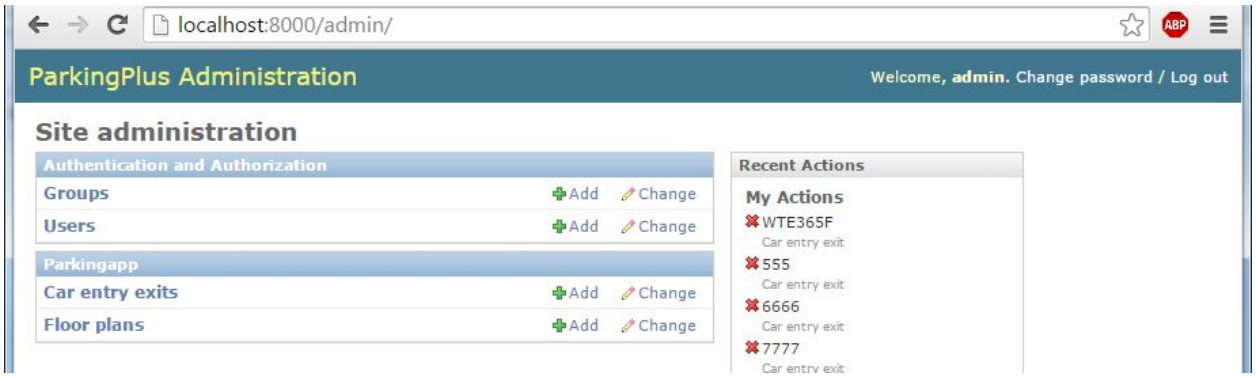
	A	B	C	D	E	F	G	H	I
1	Car No	Floor No	Entry Date	Exit DateT	Fee Paid				
2	wewe	4	2015-03-2	2015-03-2	0				
3	ereer	2	2015-03-2	2015-03-2	0				
4	qwqw	2	2015-03-2	2015-03-2	3				
5	wewewe	1	2015-03-2	2015-03-2	20				
6	aaaaa	1	2015-03-2	2015-03-2	3				
7	sffdgd	2	2015-03-2	2015-03-2	3				
8	ababab	3	2015-03-2	2015-03-2	0				
9	bzbz	3	2015-03-2	2015-03-2	3				
10	new car	2	2015-03-2	2015-03-2	3				
11	car2	2	2015-03-2	2015-03-2	3				
12									

Downloaded Excel file

4- Admin Page – click on **<u>Admin Login</u>** link on Home page-

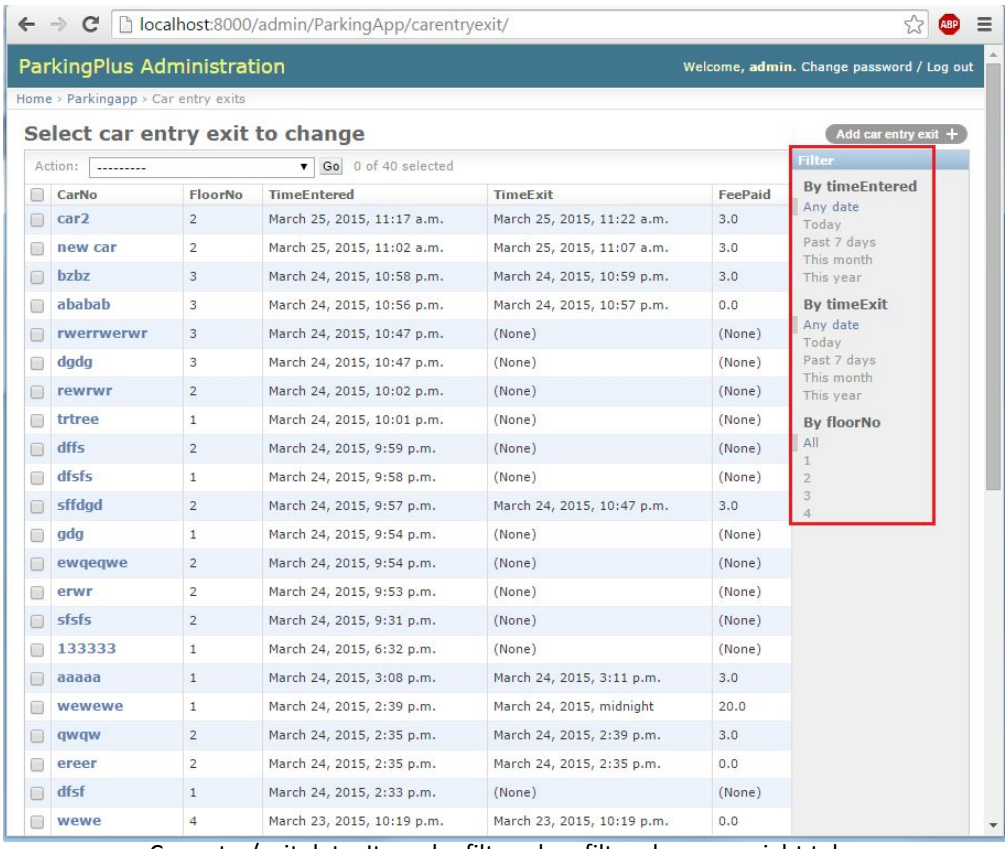


Admin Login page (username – admin, password-admin)

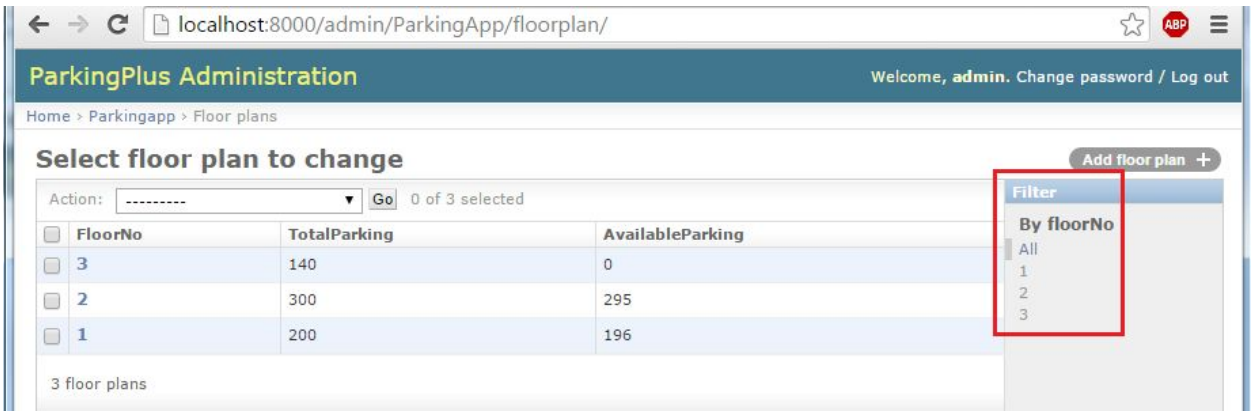


Shows DB models and user groups and actions

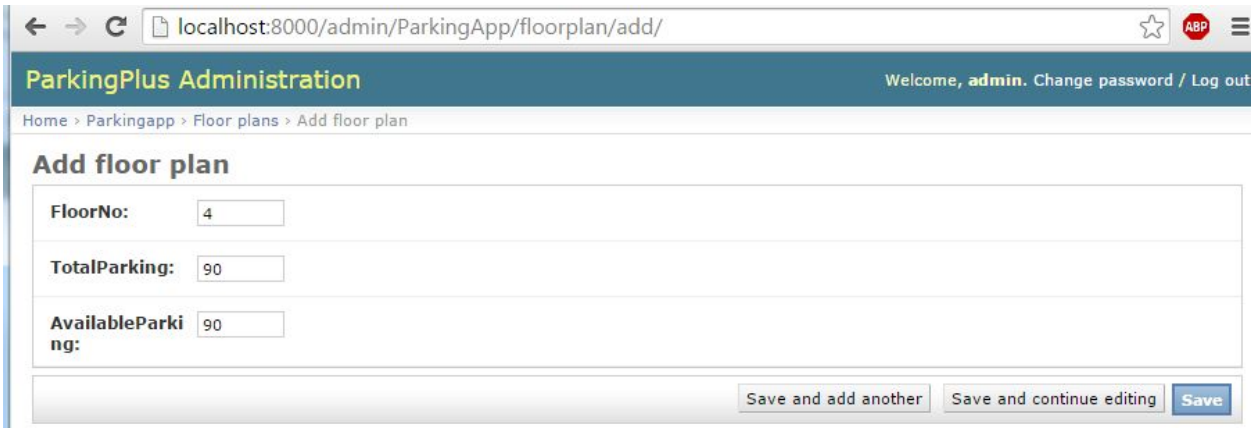
Admin users could be added here



Car entry/exit data, It can be filtered on filter shown on right tab



Floor data with the filter



Admin can add floor here

5 CONCLUSION

This ParkingPlus Application solve the parking maintenance problem to some extent. Admin knows if parking is full or available, it will save you to waste your time unnecessarily in searching parking if it is already full. It calculates the parking fee based on the time the car is parked. Currently the parking fee is fixed in program but this application could be extended to next level with variable fee based on admin input. It generates the report in excel file format to see history of car parked if required.