Rent Room Management System with Django

Arnettha Septinez (13514093)
Informatics Engineering
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
arnettha@gmail.com

***Abstract*—This paper will discuss about rent room management system. Making manual documents lacks of flexibility may cause human errors. Therefore, application-based management system would make documentations easier and more reliable. The system will be implemented using Django, a Python-based framework. The final product will be a website, including its own administrator page.**

Keywords*—*rent room; Python; django; management system; website

I. Introduction

Renting a room means paying a bedroom from someone's house for a living in a certain time period. Rent room is almost the same as apartment, except renting a room literally means only renting the bedroom. It means we share facilities like kitchen or bathroom with the other tenants. However, some of the rent rooms have a personal bathroom included. Rent room and apartment also differ in their scale. Generally, rent room has rooms open for rent less than those of apartment. While apartment is popular among families and those who are established, rent room is popular among college students, especially in Indonesia. Many college students come from cities outside the city of their colleges and have no fixed place to live. Therefore, rent room is a solution for them since rent room is relatively economical.

Nowadays, rent room is becoming more apartment-like. The 'house' used is not like a family house, but more like a flat. This kind of rent room generally has more available rooms than those of the rent room which still uses the family house architecture. It therefore results in more effort for the tenants' management. There will be more things to be monitored and more things to be documented.

At times, it is considerably troublesome to search through a long archive just to see the occupancy status of the tenants. Relatively large effort needs to be taken out to manually skim big number of documentation. Moreover, human errors can occur. Simple errors like reading the wrong tenant's name or room number can be fatal and sometimes traceback can't be done if there are too many mismatched data.

To overcome the stated problem, software-based management is needed. With the usage of software, the documentation can be done more reliably. Besides, rent room owners can monitor the tenants' status more flexibly, just by accessing it from their device. The owners don't need to bring their paper-based documentation everywhere just to make a status update for their tenants or even simply just to see their status.

To implement this solution, author will use Django as the development framework. Since Django is a Python web framework, a web-based application is the final product. The author used Django as the development tool because of its clean and pragmatical design.

There will be five sections in this paper which are introduction, literature study (contains Python and Django explanation), method of implementation, interface, and conclution of this paper.

II. Literature Study

A. *Python*

Python is a high-level programming language. It can be used for procedural, fuctional and imperative paradigm, but it is widely used for the object oriented paradigm. Python uses dynamic typing, which makes it more flexible. Therefore, deep understanding of data types need to be acquired before using Python because its dynamic typing may sometimes be confusing.

Presently (as of May 2017), Python has two available version which are Python 2 and Python 3. The main difference of Python 2 and Python 3 is their syntax. Although the syntax is not entirely different, using the wrong syntax version may cause compile error. One of the differences is on its print syntax. The example syntax to print "Hello World" is shown below for the 2.x.y version.

```
print "Hello World"
```

The 3.x.y version syntax is below.

```
print("Hello World")
```

Notice the brackets on the 3.x.y version.

In contrast of static typing, Python's dynamic typing enables its user to make a variable without explicitly declare its type. Consider the following example:

```
int a;
a = 10;
```

The above code is an example of a static typing. Before using the variable a, the variable type must be declared first. As in this case, the variable type is an integer. Then, consider the following example:

The above code is an example of dynamic typing. The variable type does not need to be declared. The variable a is automatically assigned as an integer.

B. Django

Django is a free and open source web framework which use Python as its programming language. Django follows the MTV (Model-Template-View) architectural pattern. The "model" in MTV is similar with MVC (Model-View-Controller) architectural pattern; they both represents the data that will be utilized within the system. Aside from MVC, the "view" in MTV is not how data is presented, but which data is presented. The "view" will typically render the "template" where the data is actually presented. However, views are not only used for rendering templates, a view is actually a callback function for a URL. We may define a URL for a certain function from a view, then we can call that function by calling the representative URL.

Django uses the object oriented programming paradigm. The data entities are usually represented as classes in the model. Inside the class, we can define its attributes. We can access and manipulate the data from the view by importing the classes. Below is the code example of importing class "cat" and "dog" from the model (the file name of the model is usually models.py).

```
from models import (cat, dog)
```

Here is the example for a cat model, with the attribute name which is a string, date of birth which is a date, and breed which is a string.

```
from django.db import models
class cat(models.Model):
    name = models.CharField(max_length=40)
    dob = models.DateField()
    breed = models.CharField(max_length=30)
```

The above cat model would create the following database:

```
CREATE TABLE cat (
   "id" serial NOT NULL PRIMARY KEY
   "name" varchar(40) NOT NULL
   "dob" date NOT NULL
   "breed" varchar(30) NOT NULL
);
```

Django can be implemented using Python 2 or Python 3. However, it is recommended to use Python 3, especially if we are building a new project since Python 3 is often faster, has more features, and more supported.

III. Methods

This section explains how the rent room management system will be implemented using Django. The back end will be built using Python 3, the front end will be built using HTML5 with bootstrap, and preferably using pyCharm as the IDE. The first thing we need to do is to design the database. The image below is the proposed database design.

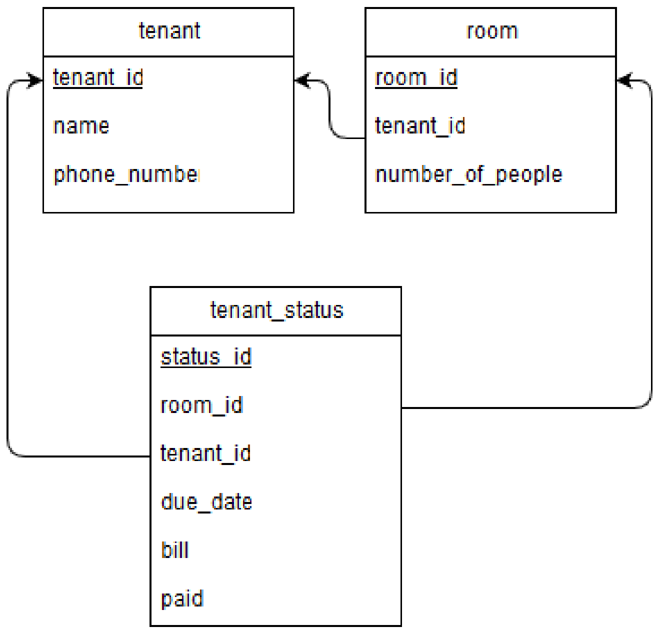


Fig. 1. Database Design

After that, we need to create a Django project. Django can create its own administrator page. To access the page, we need to create a superuser first, by using the following command in the project root directory.

```
Python manage.py createsuperuser
```

We then will be prompted to create username, e-mail address, and a strong password. The administrator page can be used for managing the data without directly accessing the database or editing the code. The page can be accessed by the URL http://<server-path>:<port(optional)>/admin/.

There will be six pages (excluding the admin page) that will be built. The pages are:

1. Login page
2. Registration page
3. Notification page
4. Tenants page
5. Edit tenant status page
6. Add new tenant page

The superuser that has been created can log in directly through the login page. However, the non-superusers have to register first. The registration require an authorization code which is defined beforehand.

The notification page displays all the tenants who haven't fully paid for the rent near the due date. The tenants page displays the list of all tenants. The details of the tenants are editable. The owner of the rent room can also add new tenants or delete existing tenants.

IV. Interface

According to earlier section, there are six pages for the system. Below is the designed interface for the pages.

1. Login page

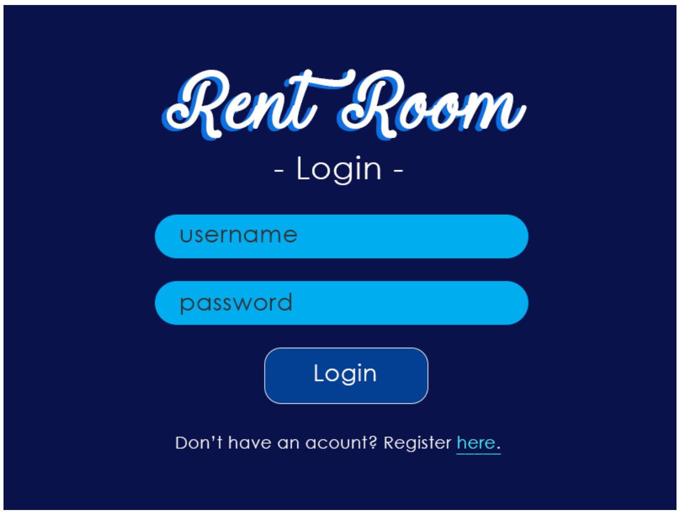


Fig. 2. Login Page

2. Registration page

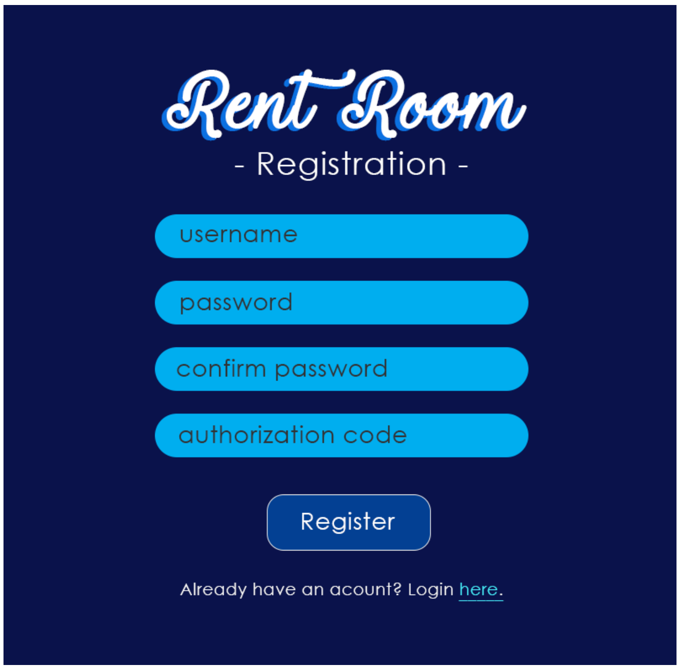
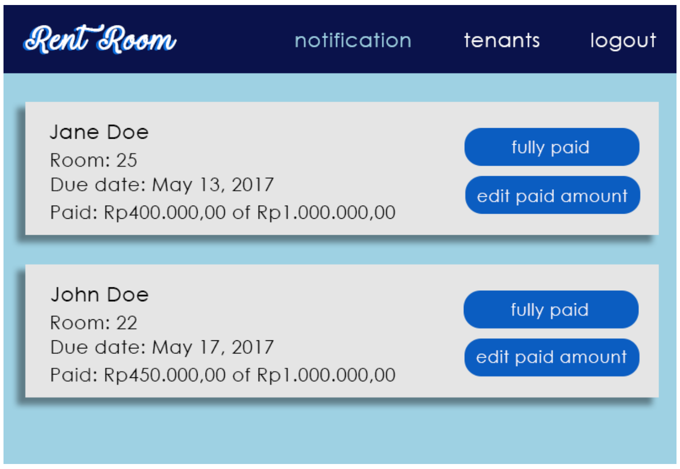


Fig. 3. Registration Page

3. Notification page



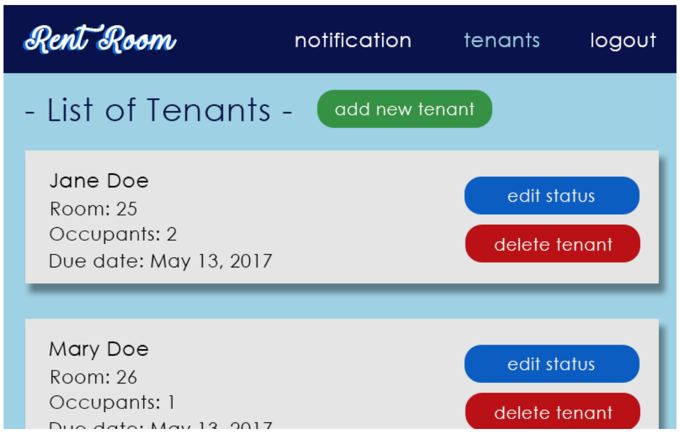
Jane Doe
Room: 25
Due date: May 13, 2017
Paid: Rp400.000,00 of Rp1.000.000,00

John Doe
Room: 22
Due date: May 17, 2017
Paid: Rp450.000,00 of Rp1.000.000,00

Fig. 4. Notification Page

If the user clicks the "edit paid amount" button, the paid field will be changed into a textfield. The user will be able to change the paid amount. If the user clicks "fully paid" button, the relevant rent will be considered paid off and will be deleted from notification. The due date will be reset afterwards.

4. Tenants page

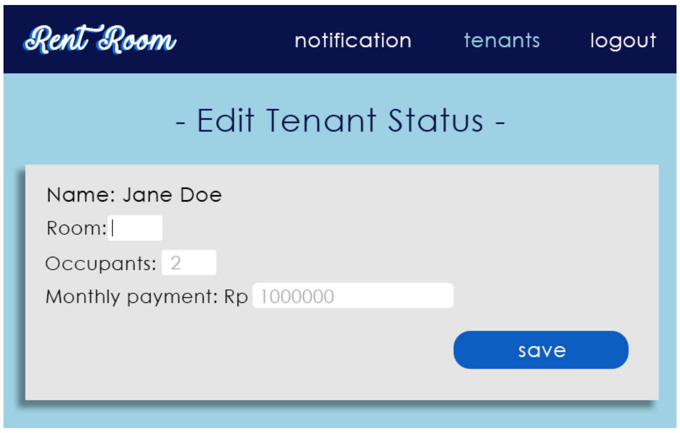


Jane Doe
Room: 25
Occupants: 2
Due date: May 13, 2017

Mary Doe
Room: 26
Occupants: 1
Due date: May 13, 2017

Fig. 5. Tenants Page

5. Edit tenant status page



Name: Jane Doe
Room: |
Occupants: 2
Monthly payment: Rp 1000000

Fig. 6. Edit Tenant Status Page

6. Add new tenant page

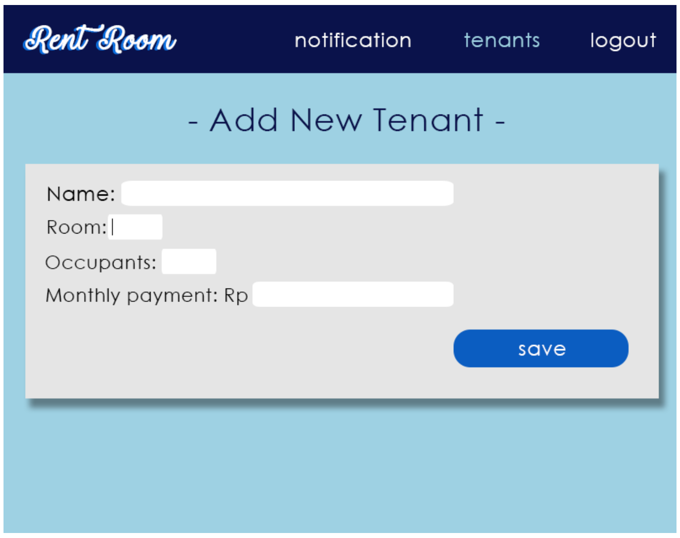


Fig. 7. Add New Tenant Page

V. Conclusion

Django is designed to build a website rapidly. Therefore, it is suitable for making a simple web application like rent room management system. Django's MTV architectural pattern supports the system's work cycle which mostly involves data presentation and manipulation. Django also supports high scalability of its products. With that feature, extension of the application won't be a hustle (e.g. adding new house for rent room).

Acknowledgment

I'm fully thankful for IF3280's lecturers, Dr. Dessi Puji Lestari, Dr. Ir. Rinaldi Munir, MT., and Dr. Eng. Ayu Purwarianti, ST., MT. for every priceless knowledge and wisdom that had been shared with all IF3280 students. I'm also thankful for my dearest friends who support me and be the source of my courage while writing this paper.

References

[1] IF3280: Komunitasi Tulisan Presentation Slide
Accessed on May 4, 2017 17:24

[2] https://www.python.org/
Accessed on May 4, 2017 20:35

[3] https://www.djangoproject.com/
Accessed on May 5, 2017 12:11

[4] https://www.sitepoint.com/typing-versus-dynamic-typing/
Accessed on May 5, 2017 13.10

Statement

With this, I acknowledge that this paper is my own writing, not an adaptation nor a translated version of another paper, and not a plagiarism.

Bandung, May 5, 2017

[illegible]

Arnettha Septinez (135141093)