|  |  |
| --- | --- |
| **Release:** | **Document Status: Final** |
| **Version: 1.0.0** |  |

Notice Connect

tECHNICAL CHALLENGE DOCUMENT

**Created By: Sonakshi Karkera**

Table of Contents

[1. Objective 2](#_Toc62844860)

[1.1 Method used 3](#_Toc62844861)

[1.2 Limitation 3](#_Toc62844862)

[1.3 Scope and Deliverables 3](#_Toc62844863)

[2. Technical Specifications 4](#_Toc62844864)

[2.1 Software Interface 4](#_Toc62844865)

[3. Technical Design Diagram 5](#_Toc62844866)

[3.1 Web Application 5](#_Toc62844867)

[3.2 Model Details 5](#_Toc62844869)

[3.3 Views and App logic Details 5](#_Toc62844870)

[4. Database 6](#_Toc62844871)

[4.1 MYSQL Credentials: 6](#_Toc62844872)

[4.2 Diagram 7](#_Toc62844873)

[4.1 Table Details 7](#_Toc62844874)

[5 Design & Implementation 7](#_Toc62844875)

[5.1 Create NoticeConnect Project and Adding Notice, Record and Match Apps 7](#_Toc62844876)

[5.2 Create tables named **Notice**, **Record**, **Match** 8](#_Toc62844877)

[5.3 Load **Notice** table with some dummy data 10](#_Toc62844878)

[5.4 Create Django Views to Create Record 10](#_Toc62844879)

[5.5 Create Django view to Delete records 12](#_Toc62844880)

[5.6 Create Django view for Listing matches 13](#_Toc62844881)

[5.7 Create Django view for Deleting Notices 14](#_Toc62844882)

[5.8 Matching Algorithm 15](#_Toc62844883)

[6. Unit Tests 16](#_Toc62844884)

[6.1 Records 16](#_Toc62844885)

[6.1.1 Test\_models.py 16](#_Toc62844886)

[6.1.2 Test\_url.py 17](#_Toc62844887)

[6.1.3 Test\_views.py 17](#_Toc62844888)

[6.2 Notices 20](#_Toc62844889)

[6.2.1 Test\_models.py 20](#_Toc62844890)

[6.2.1 Test\_urls.py 21](#_Toc62844891)

[6.2.1 Test\_views.py 21](#_Toc62844892)

[6.3 Matches 22](#_Toc62844893)

[6.3.1 Test\_models.py 22](#_Toc62844894)

[6.3.2 Test\_urls.py 23](#_Toc62844895)

[6.3.3 Test\_views.py 24](#_Toc62844896)

# 1. Objective

Objective of this technical challenge is to create fully functional web application project using Django framework. This website has three major apps: Notice, Match, and Record. The whole project named NoticeConnect is designed and implemented using Django Framework which is an excellent open-source web application frame work for complex data-driven website development. The major part of this report will introduce I used Django to create a database table, web page user interface and inner logic to handle requests to create Notice and Records. Based on the records created by the user a matching algorithm is designed which will run to categorize the records as either- Strong, Possible, Weak or No match.

### 1.1 Method used

Our Notice management system consists of three components which are Notice, Match, and Record. Each component contains three parts. I have first designed the model of the relative component for data architecture, then the template for user interface, at last implemented the view which includes all the functions.

### 1.2 Limitation

The system may have some potential bugs or flaws because of the development time constraint. However, because of the flexibility and powerful functionality of Django, these bugs or flaws can be fixed.

### 1.3 Scope and Deliverables

* Create a Django project
* Connect with any SQL database of your choice
* Create tables named **Notice, Record, Match**. Those will have following columns.

|  |  |  |
| --- | --- | --- |
| **Notice** | **Record** | **Match** |
| first\_name - required | first\_name - required | record (F.K. to Record) |
| last\_name - required | last\_name - required | notice (F.K. to Notice) |
| alt\_last\_name | province | type (ChoiceField) |
| Province - required | date\_of\_birth (DateField) |  |
| date\_of\_birth (DateField) |  |  |

* Load **Notice** table with some dummy data.
* Create Django Views to Create and Delete Record.
* Create Django View for listing matches.
* Upon record creation, record’s data will be matched against all notices. Matching algorithm is described below.

**Match Algorithm**:

1. Strong Match:

- first\_name or alt\_first\_name of notice matches with first\_name of record  
 - last\_name or alt\_last\_name of notice matches with last\_name of record

- date\_of\_birth and province are matching

2. Possible Match

- first\_name, last\_name, province are matching

3. Weak Match

- Only first\_name and last\_name are matching

* If match found, create Match object.
* Related matches needs to be deleted when Notice or Record is deleted.

# 2. Technical Specifications

## 2.1 [Software](#_chndllrje83z) Interface

* **Information Systems Infrastructure Databases**: MS SQL, ORACLE, MySQL
* **Development IDE**: MS Visual Studio Code, MySQL Workbench (Windows)
* **Programing languages and frameworks**: Django, Python, Django, MySQL, JavaScript, HTML
* **Database server:** AWS RDS Instance
* **Admin**
  + Django 3.1.5
* **UI:** framework Django Bootstrap and CSS.
  + django-bootstrap-form Version 3.4
  + django-bootstrap-modal-forms Version 1.5.0
  + django-bootstrap-navbar Version 0.0.20

# 3. Technical Design Diagram

### 3.1 Web Application

### 

localhost

### 3.2 Model Details

These are used to create database table instances to retrieve, update or insert data from the below tables

* Notice
* Record
* Match

### 3.3 Views and App logic Details

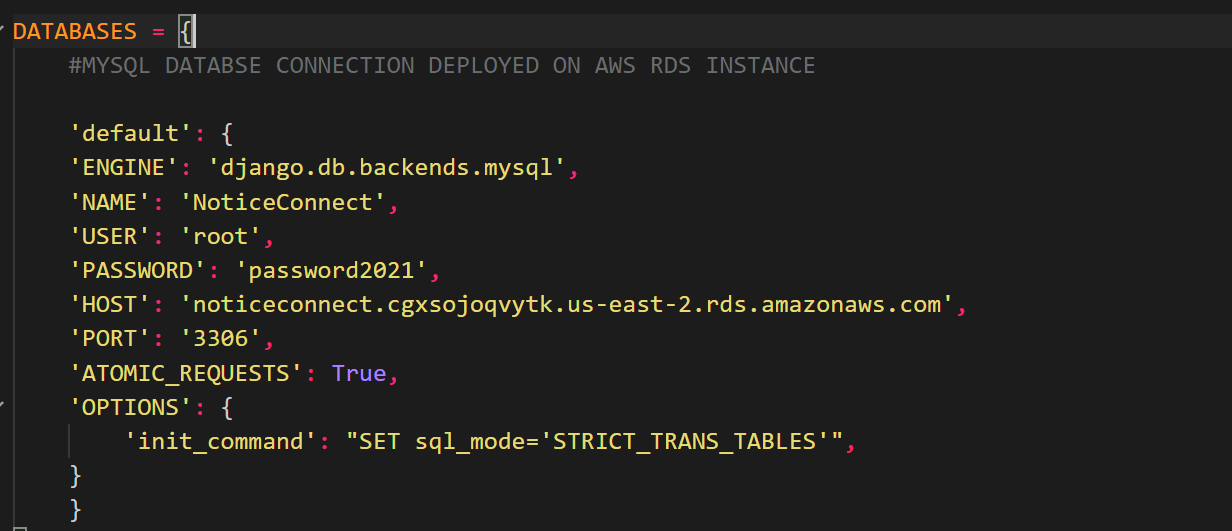
Functions and class.

* Match Algorithm
* Create Record
* List Record
* Delete Record
* List Match
* List Notice
* Delete Notice

# 4. Database

For purpose of this project, instead of using default sqlite3 database I have used MySQL database. I have hosted this database on AWS RDS instance. Attached below screenshot displaying credentials and ER diagram retrieved from the database brief overview

### 4.1 MYSQL Credentials:

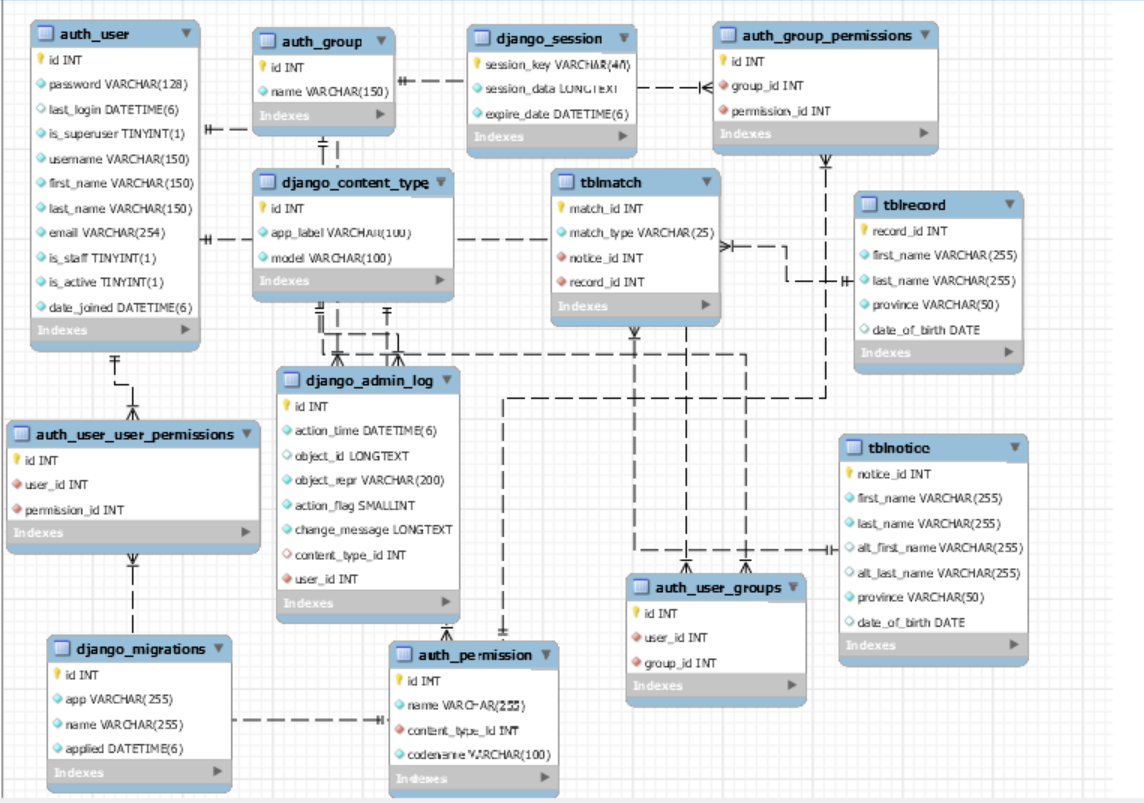


Admin User Credentials:

Username :root

Password :password2021

### 4.2 Diagram



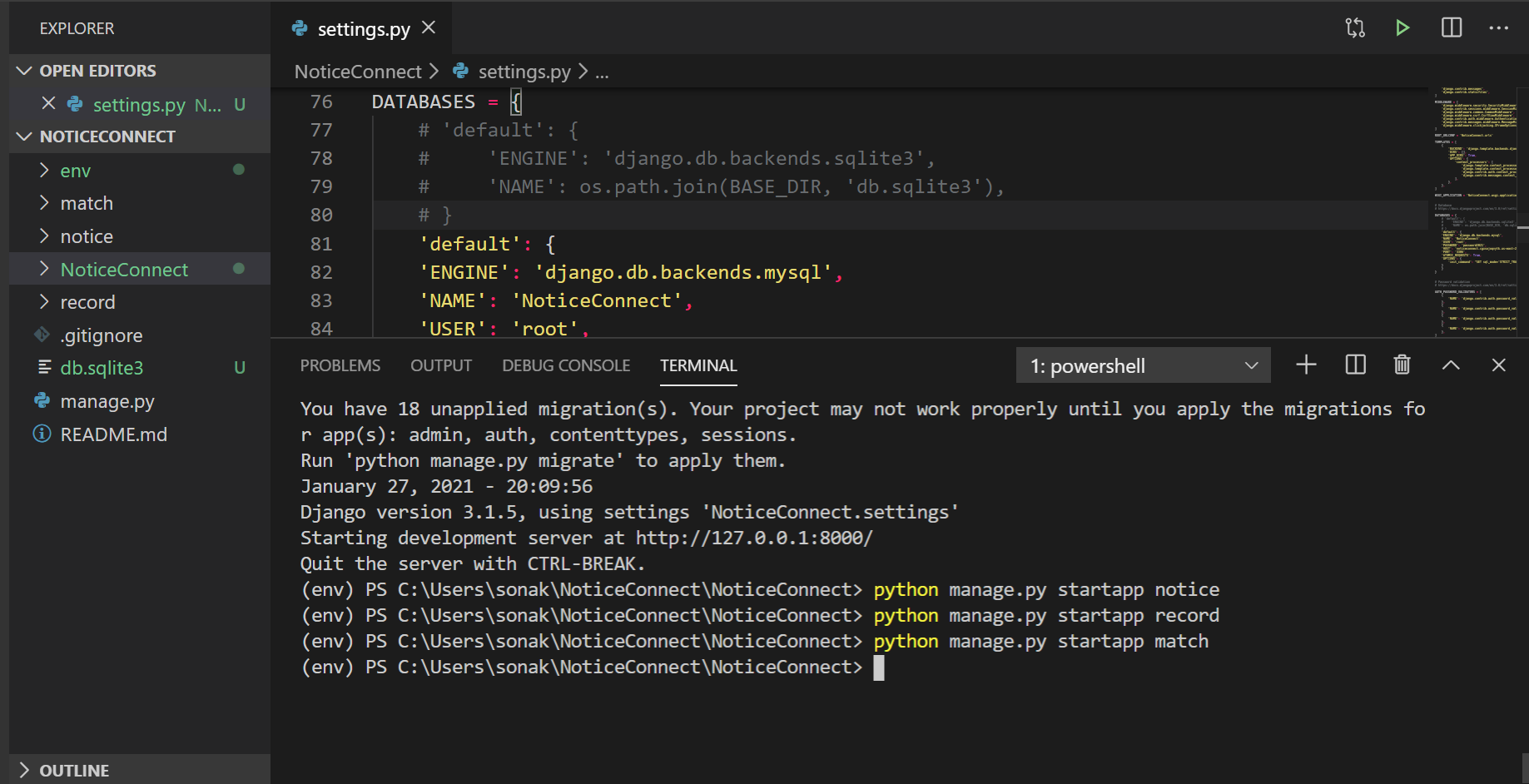
### 4.1 Table Details

1. tblnotice– notice\_id,first\_name,last\_name,alt\_first\_name,alt\_last\_name,province,date\_of\_birth
2. tblrecord – record\_id, first\_name,last\_name, province, date\_of\_birth
3. tblmatch – match\_id record\_id, notice\_id,match\_type

# 5 Design & Implementation

### 5.1 Create NoticeConnect Project and Adding Notice, Record and Match Apps

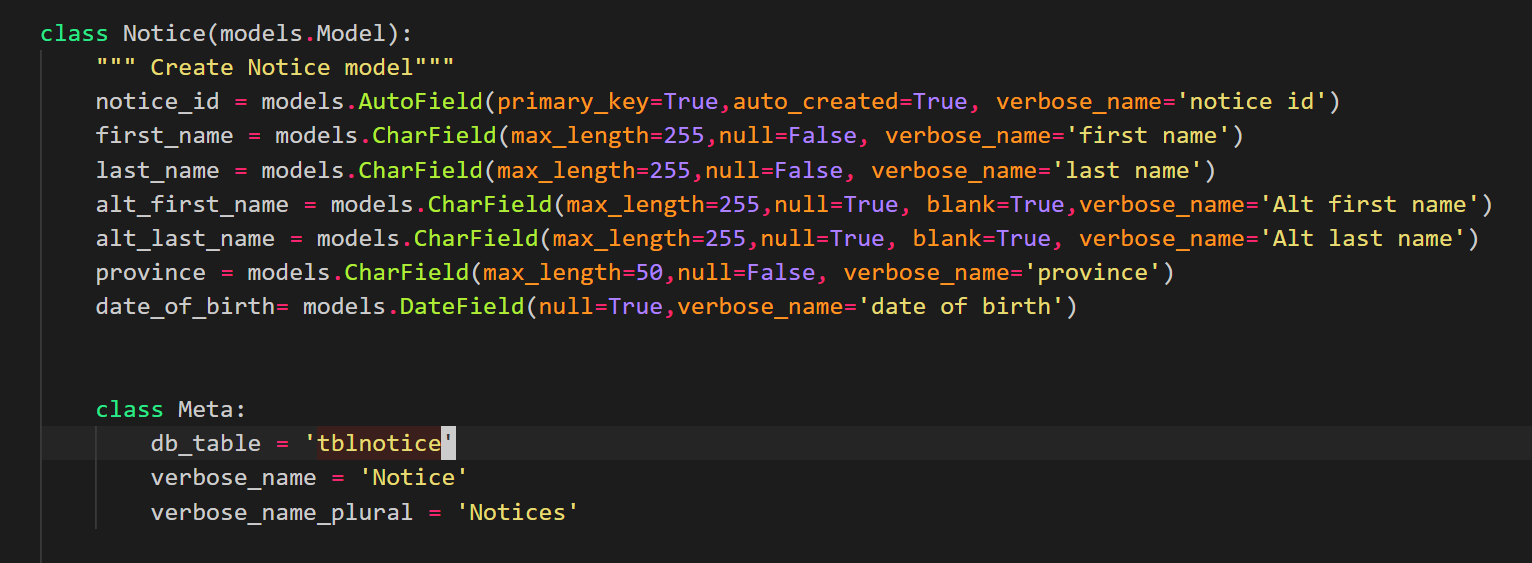
Main project is named Notice Connect and apps have been created as seen in the screenshot below:



### Create tables named **Notice**, **Record**, **Match**

Next, we create tables in the database. Django uses a class called model to represent the database table schema. To create table, we just need to write a new class derived from Django model class and later migrate it. On migration Django will run the SQL representation scripts at the back and create tables in the database.

* + 1. Notice Models.py



* + 1. Match Models.py

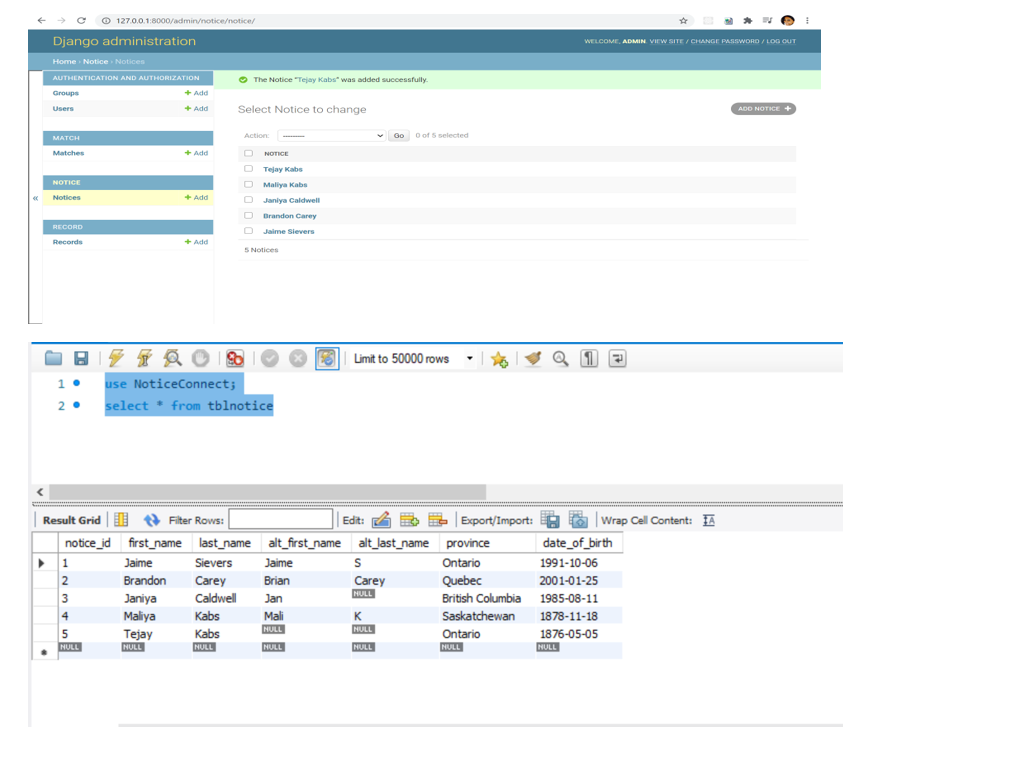


* + 1. Records Models.py



### Load **Notice** table with some dummy data

To load the Notice table with dummy table I simply registered notice model in admin.py and used admin representation web interface to add dummy records. Below is the screenshot of the records.



### Create Django Views to Create Record

In order to create and delete records I created a user interface to let users interact with the data indirectly. Django provides the template component to create the user interface for users. I have used Django forms as and when required.

* + 1. UI for creating records

|  |
| --- |
| **REQUEST** |
| GET <https://localhost:8000/>  Accept: application/json |
|  |

* + 1. View for create records

|  |
| --- |
| As mentioned in challenge problem statement and my understanding based of that, whenever the user entrees the details for records, at the back record will be create and along with that it will run the matching algorithm to check of the conditions meets and based of that, it will create a entry in match table. |
|  |

### Create Django view to Delete records

* + 1. UI for Delete records

|  |
| --- |
| GET <https://localhost:8000/listrecords/>  Accept: application/json |
|  |
| On click of the delete button will delete the record from the table. Additionally it will also delete the record from match table incase a match case was inserted |

* + 1. View for Delete records

|  |
| --- |
|  |

### Create Django view for Listing matches

* + 1. UI for Listing matches

|  |
| --- |
| GET <https://localhost:8000/listmatches/>  Accept: application/json |
|  |

* + 1. View for Listing matches

|  |
| --- |
|  |

### Create Django view for Deleting Notices

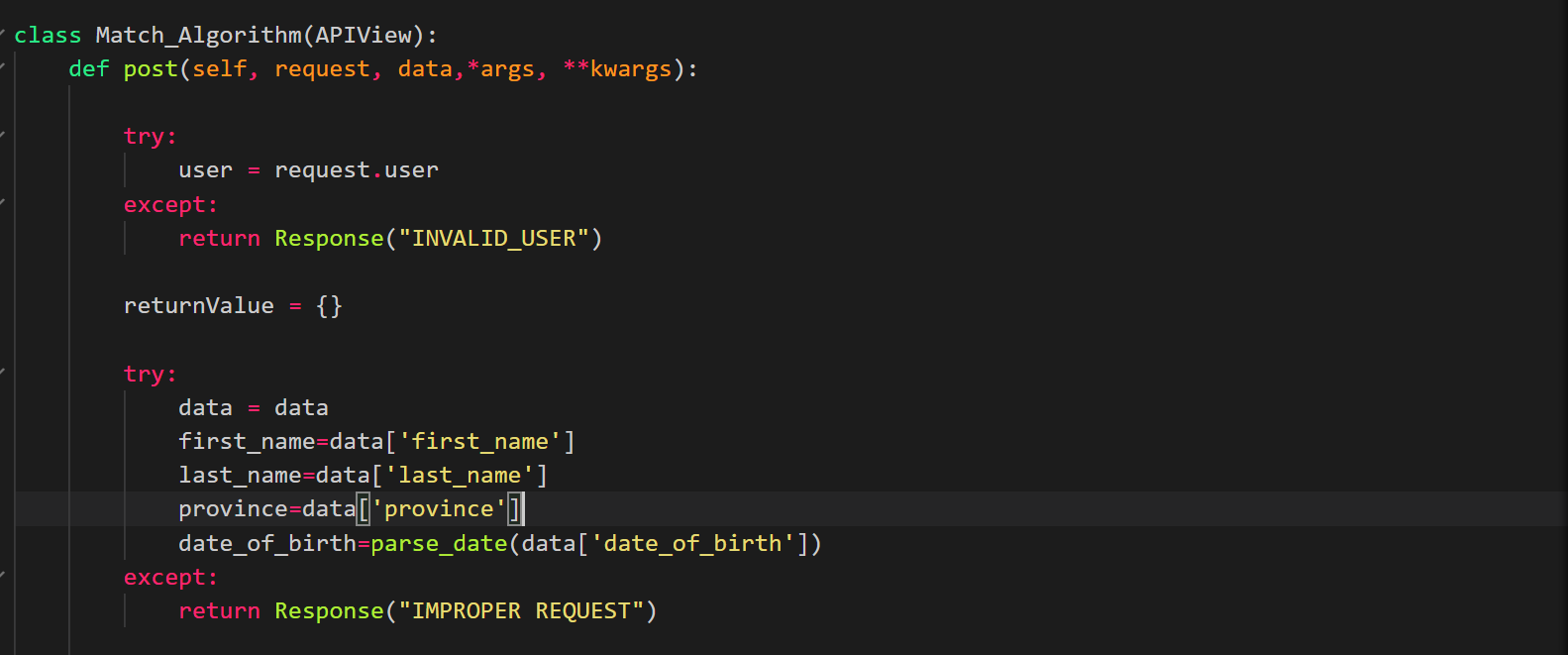
* + 1. UI for Deleting Notices

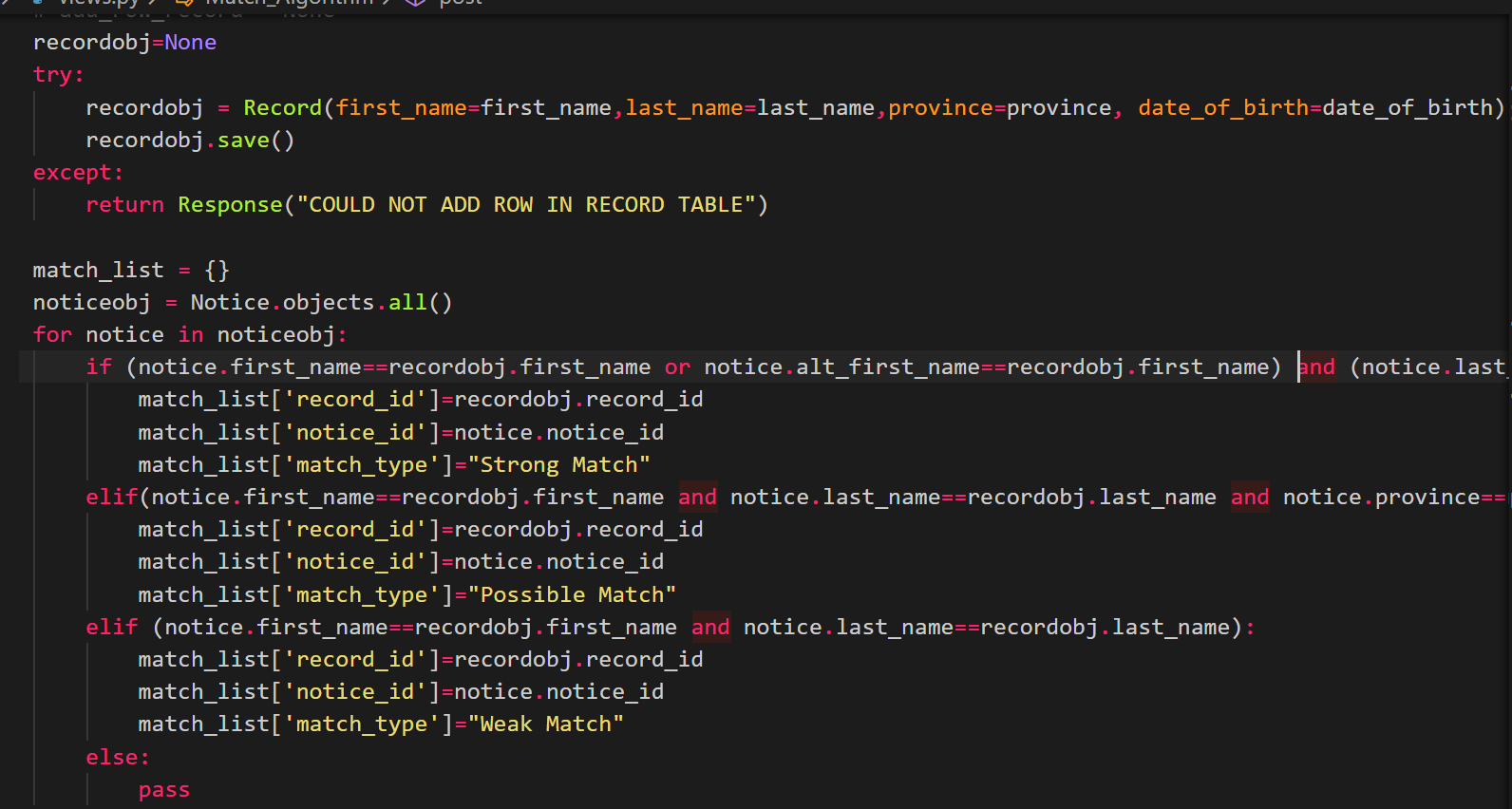
|  |
| --- |
| GET <https://localhost:8000/listnotices/>  Accept: application/json |
|  |

* + 1. View for Deleting Notices

|  |
| --- |
|  |

### Matching Algorithm



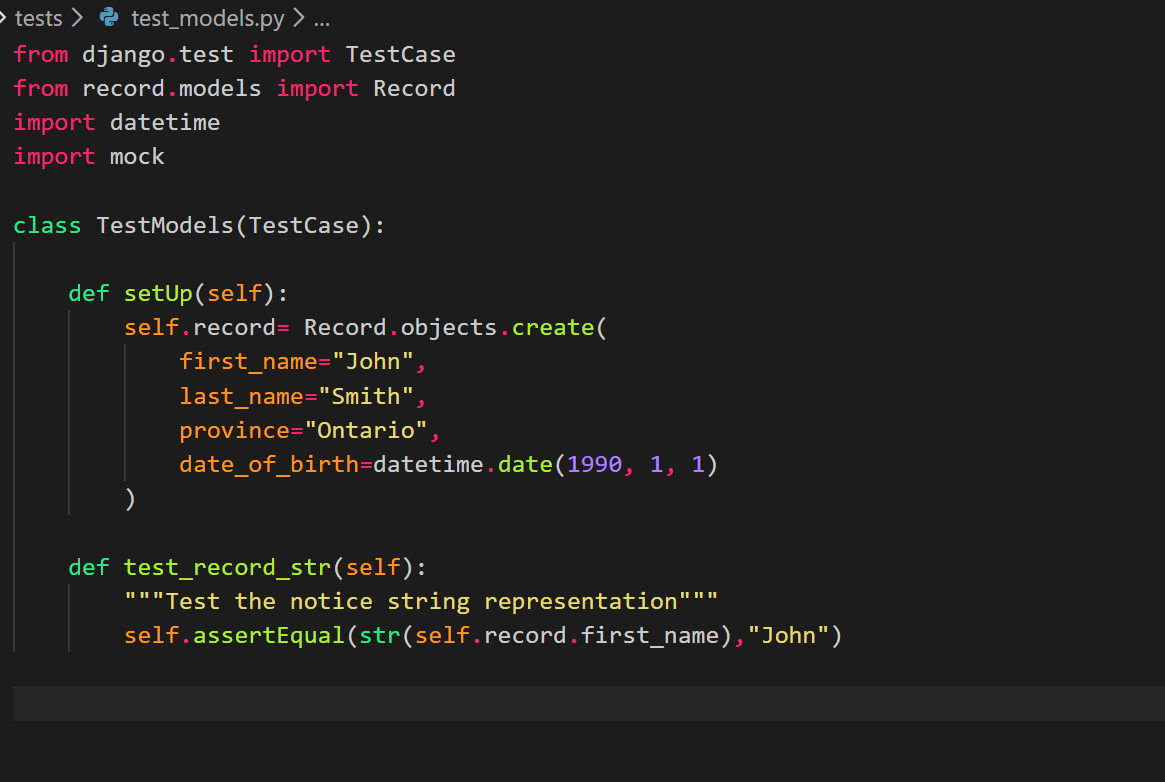




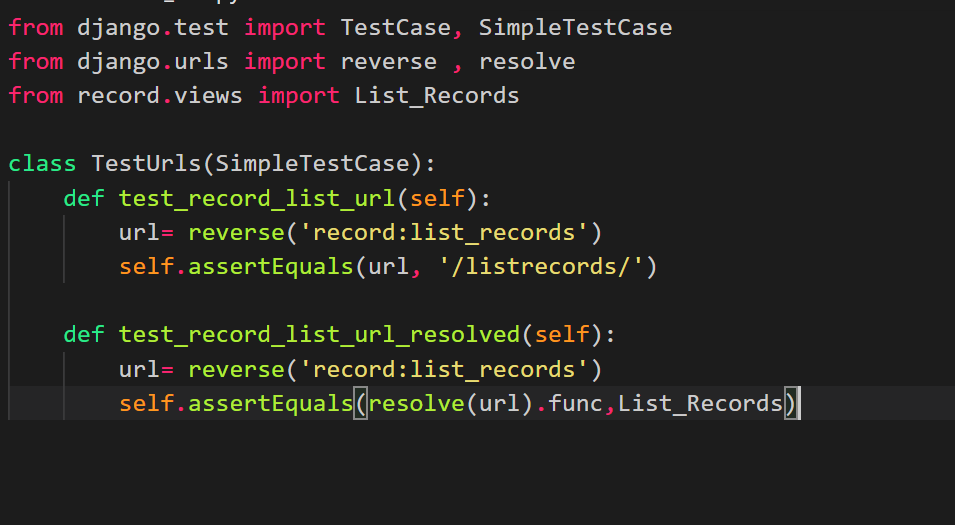
# 6. Unit Tests

## 6.1 Records

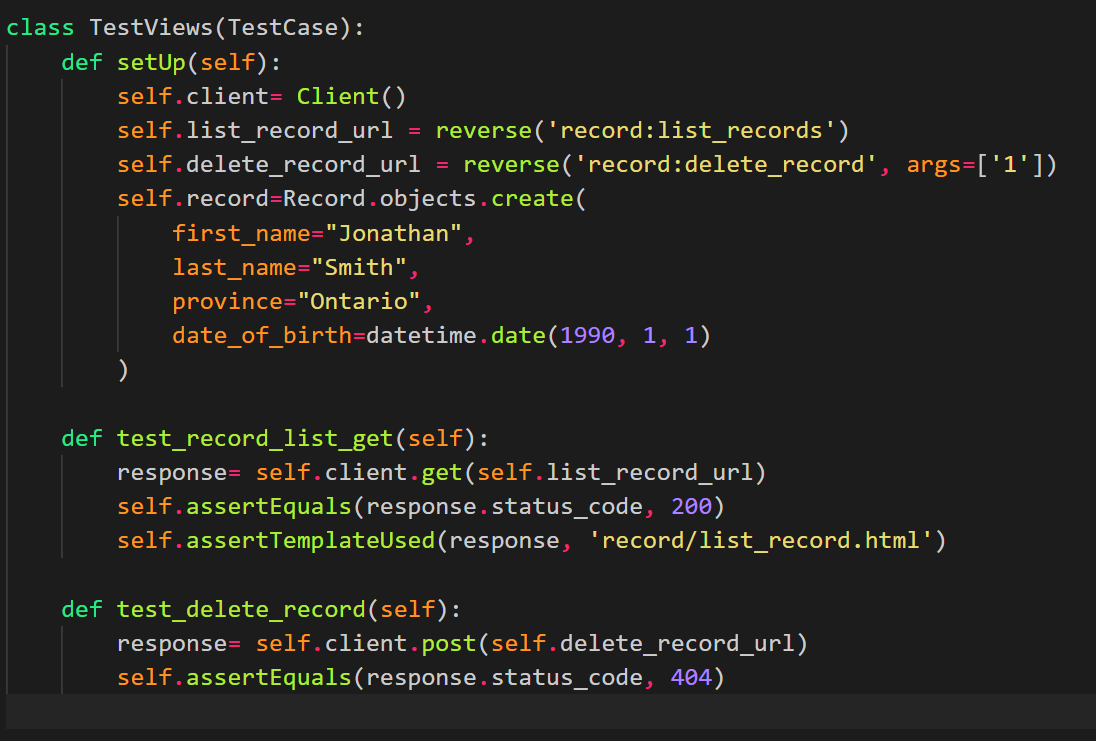
### 6.1.1 Test\_models.py



### 6.1.2 Test\_url.py

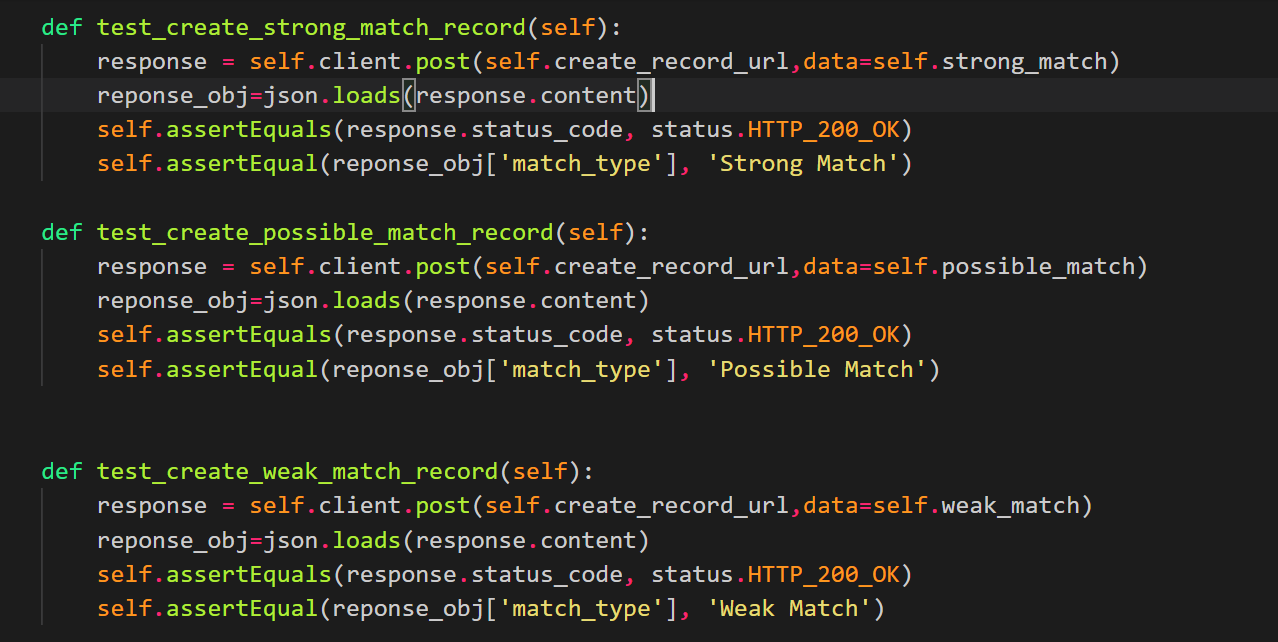


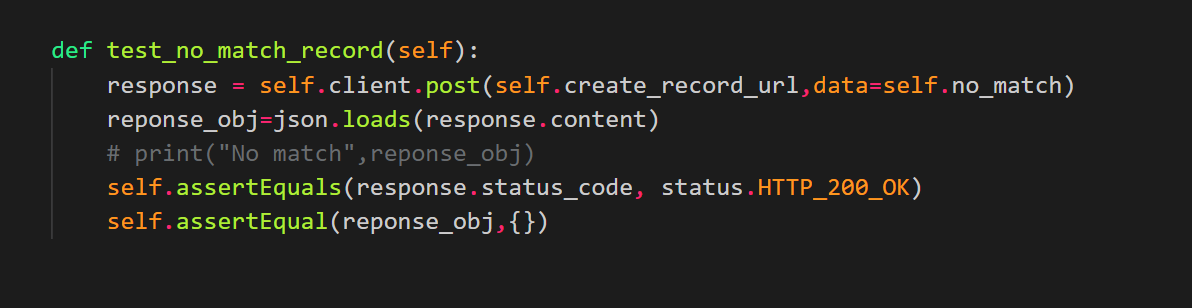
### 6.1.3 Test\_views.py

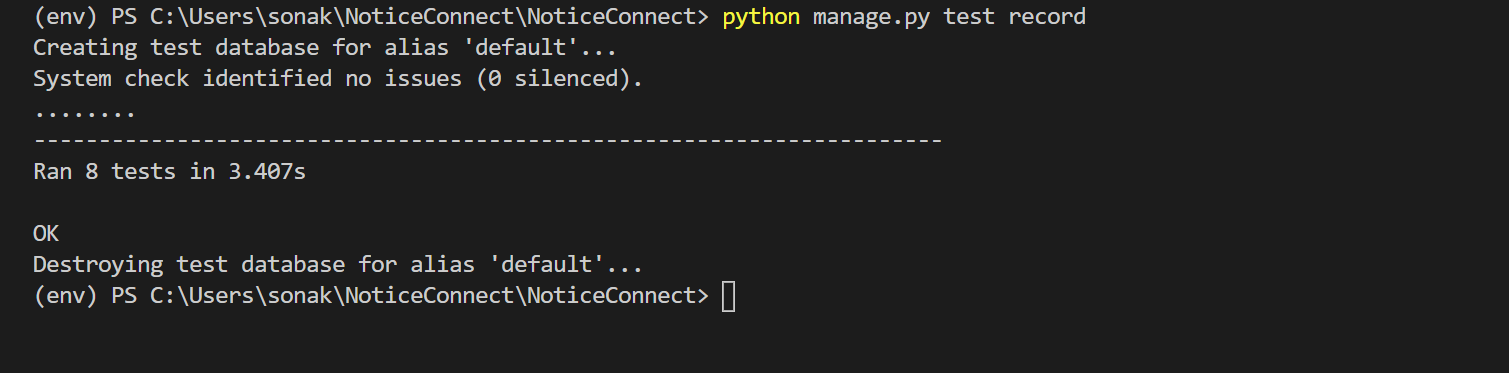






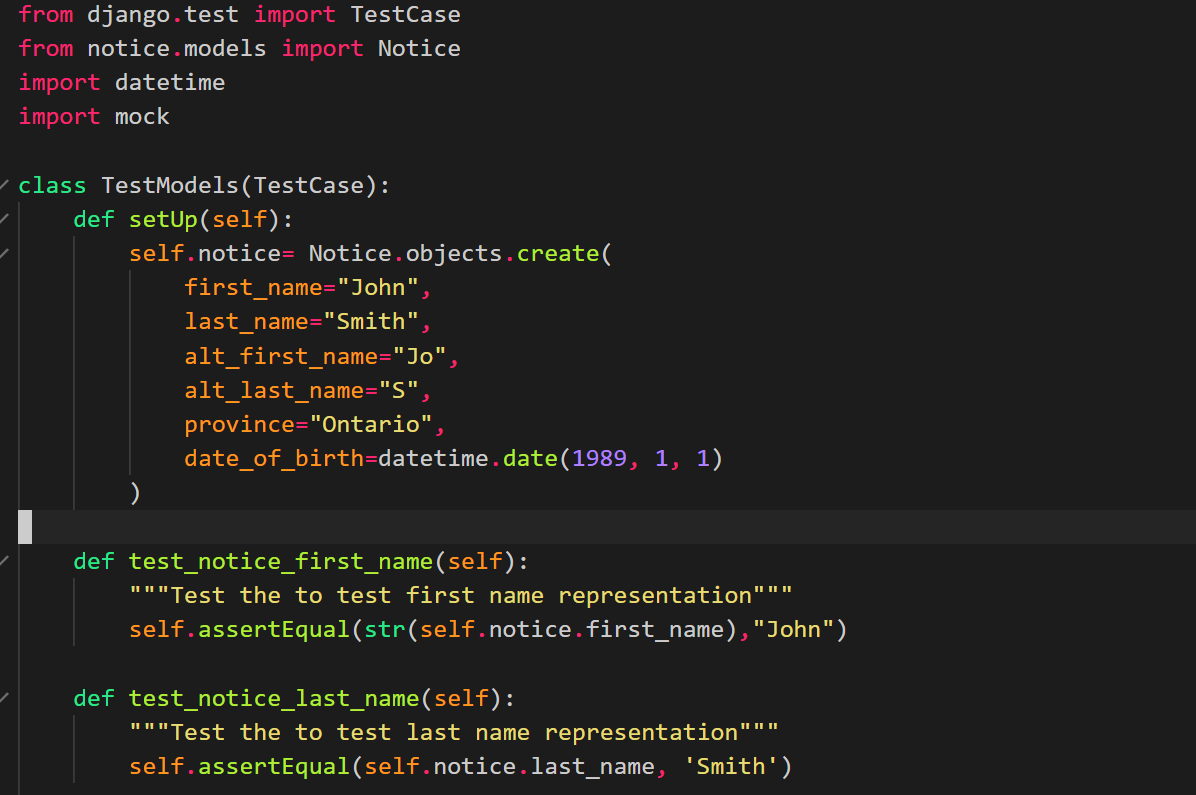


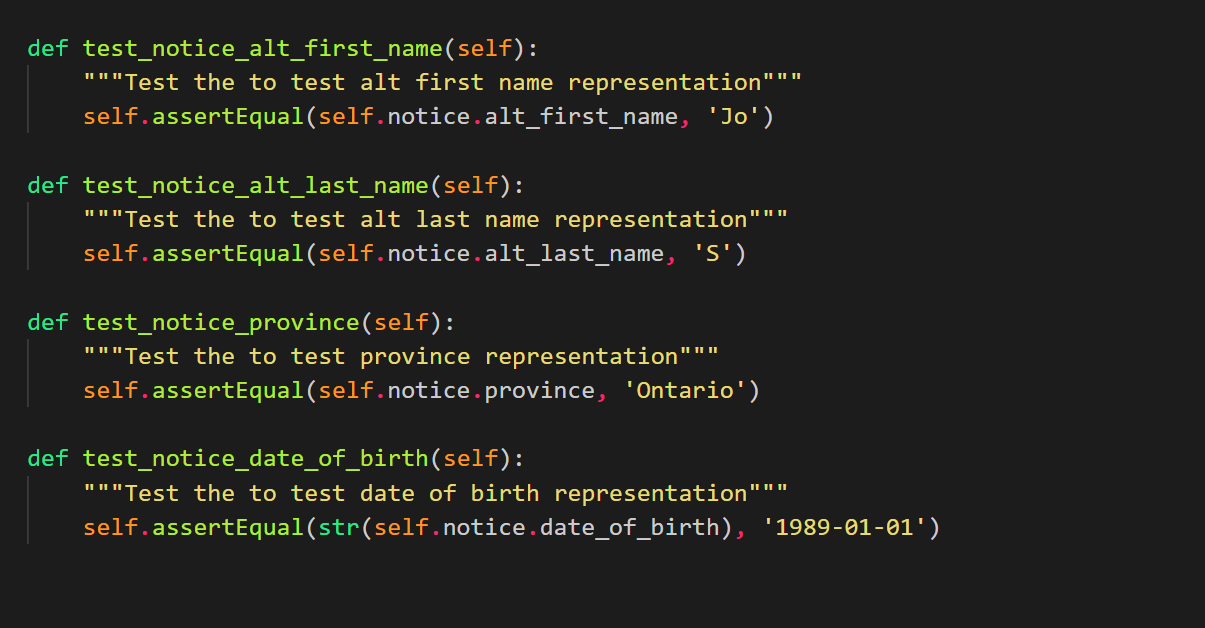




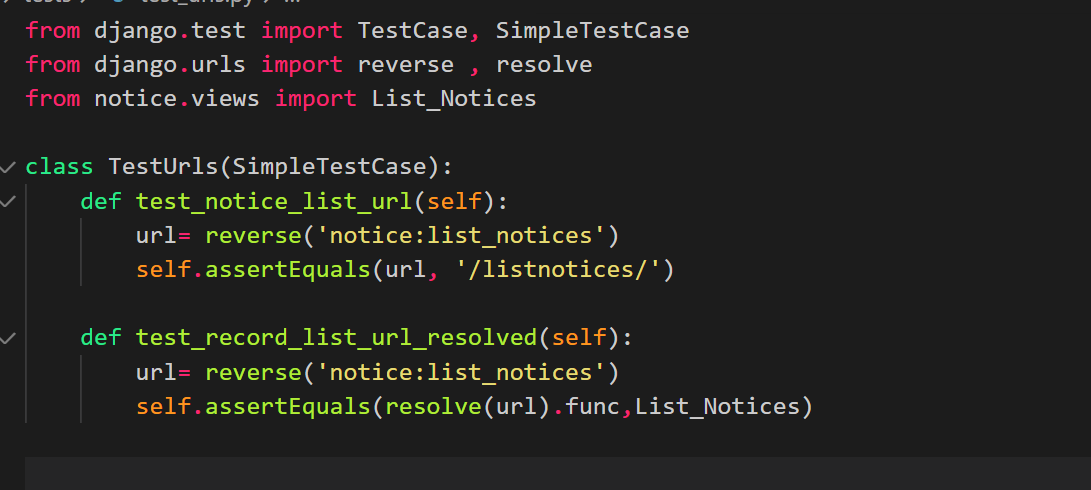
## 6.2 Notices

### 6.2.1 Test\_models.py

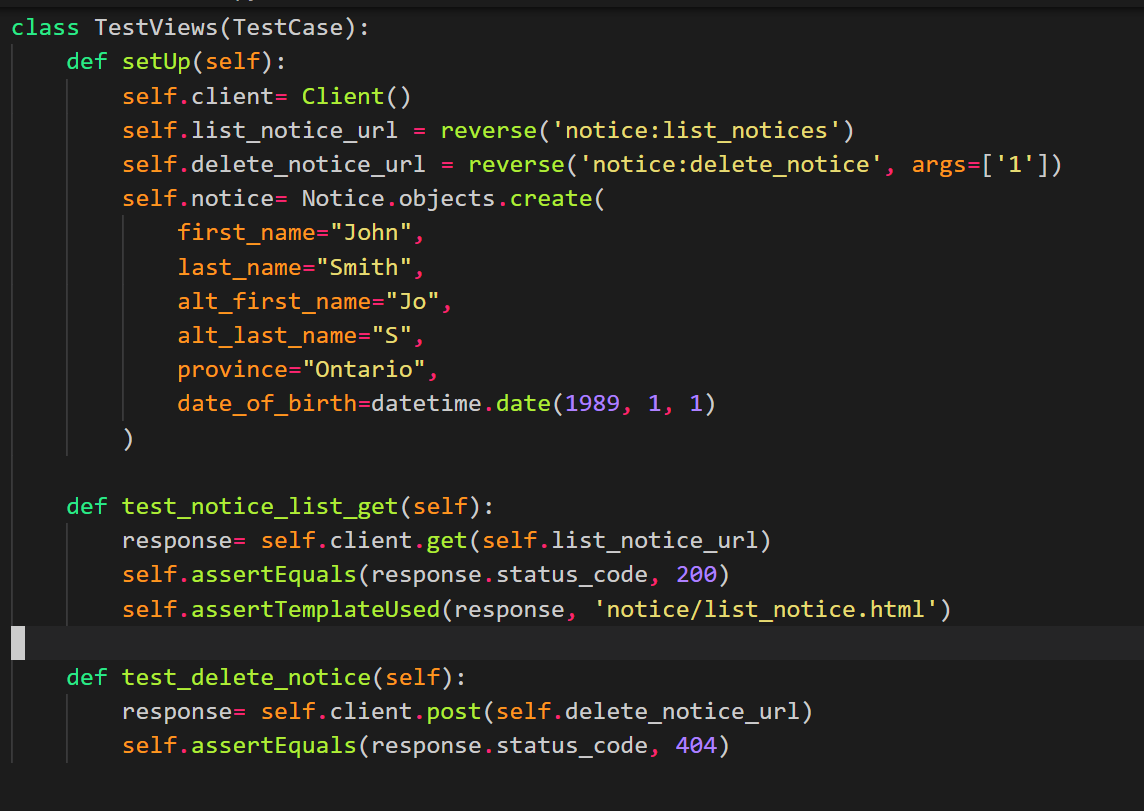


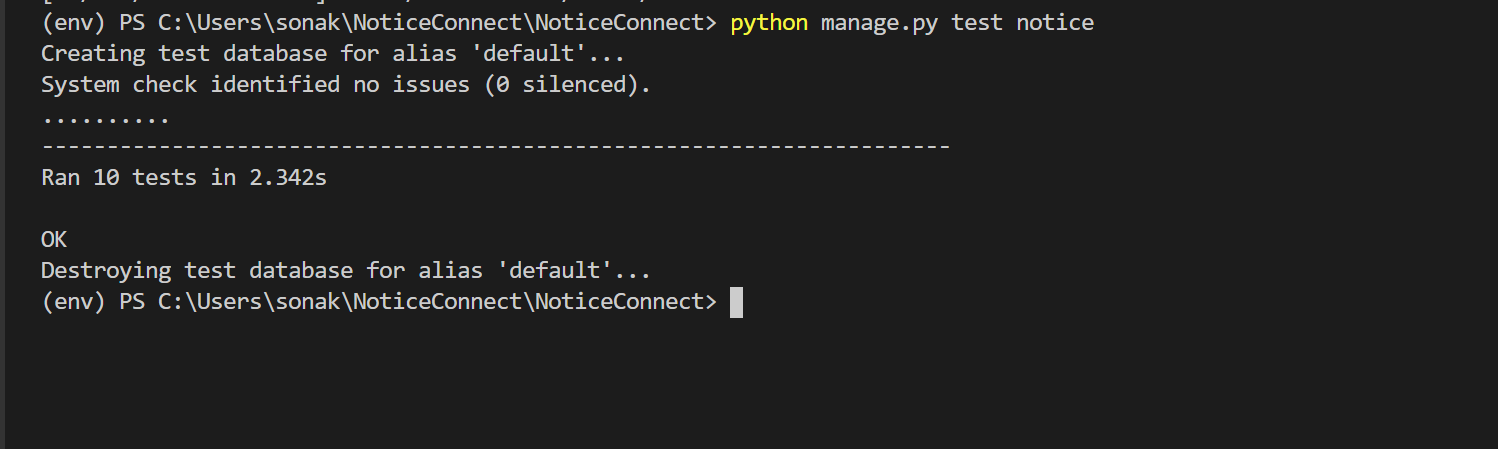


### 6.2.1 Test\_urls.py



### 6.2.1 Test\_views.py

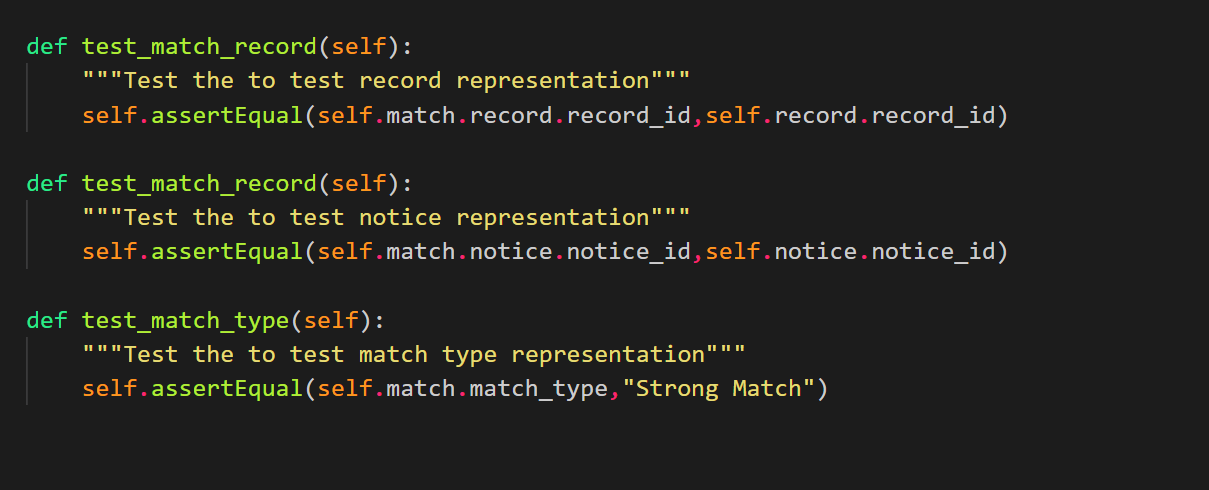




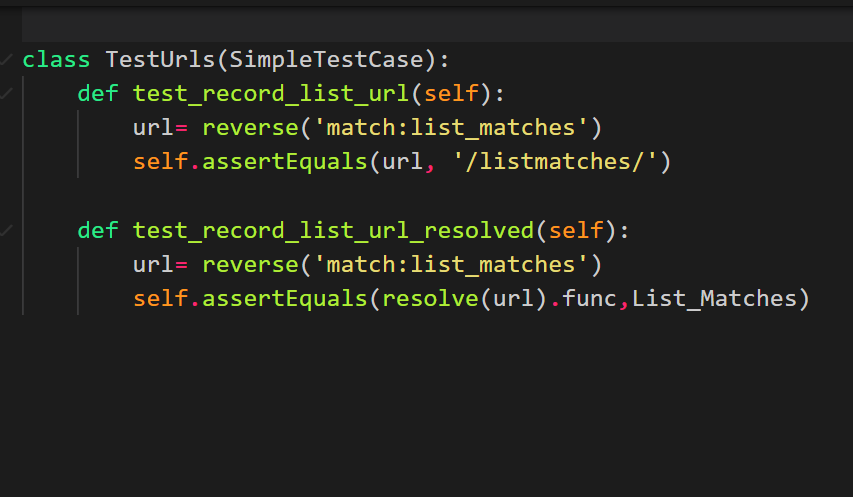
## 6.3 Matches

### 6.3.1 Test\_models.py



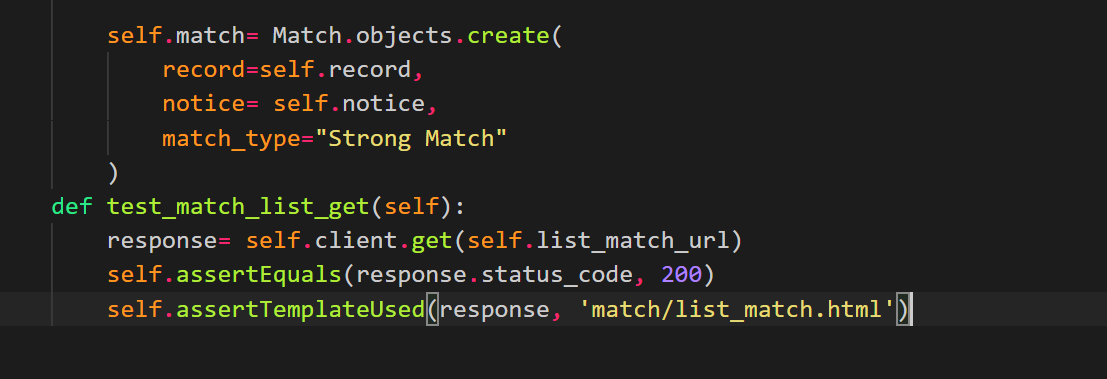


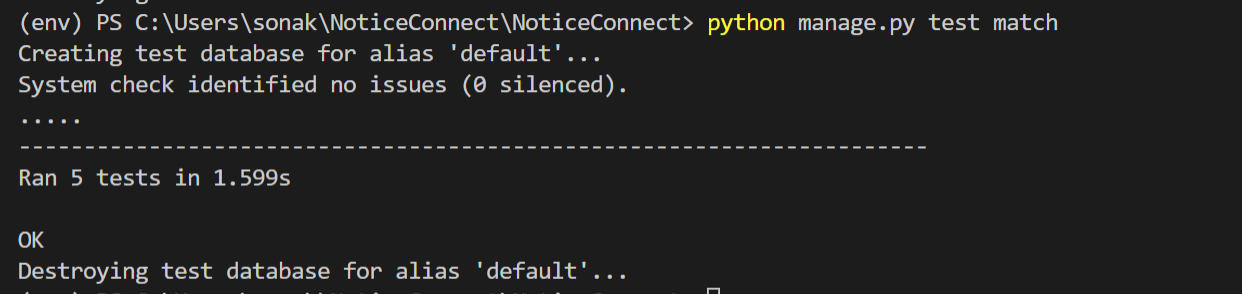
### 6.3.2 Test\_urls.py



### 6.3.3 Test\_views.py







AF