# Canteen Management System

**Group Name: Group 10**

**Group Members:**

| First name | Last Name | Student number |
|---|---|---|
| Augustine | Jose | C0753676 |
| Ashish Kumar | Chaudhaari | C0755148 |
| Dhaval | Vakil | C0752581 |
| Nirmala | Shayamala | C0753977 |
| Vineetha | Shaily | C0752287 |

**Submission date:** **17/08/2020**

## Contents

## Abstract

- ➢ Nowadays people don’t have much time to spend in canteen.
- ➢ Many customer visits the canteen in their lunch break and recess so they have limited time to eat and return to their respective office and colleges.
- ➢ Manual system involves paper work in the form of maintaining various files and manuals.
- ➢ Maintaining critical information in the files and manuals is full of risk and a tedious process.

## EXISTING SYSTEM

- ➢ Manual systems
- ➢ Queues for ordering food
- ➢ Maintaining Paper Based Records of sales

## PROPOSED SYSTEM

- ➢ Smart Canteen is a Progressive Web Application that it greatly simplifies the ordering process for both the customer and the canteen.
    - Online Ordering
    - View daily, weekly and monthly sales
    - Predictions of food items in high demand
    - Up-to-date Menu
    - Online Payments
    - Ratings and Feedback

### SCOPE

- ➢ This system requires very fewer time factors as compared to manual system.
- ➢ It provides fast and efficient automated environment instead of slow error prone manual systems , thus reducing time and manpower.
- ➢ Easy to use GUI.
- ➢ All the sales and profits information will be stored on the system providing backup and records.

## System Requirements

### Hardware Requirements

| | |
|---|---|
| RAM | 4 GB Minimum |
| Processor | i3 Minimum |
| Hard disk | 500 GB |

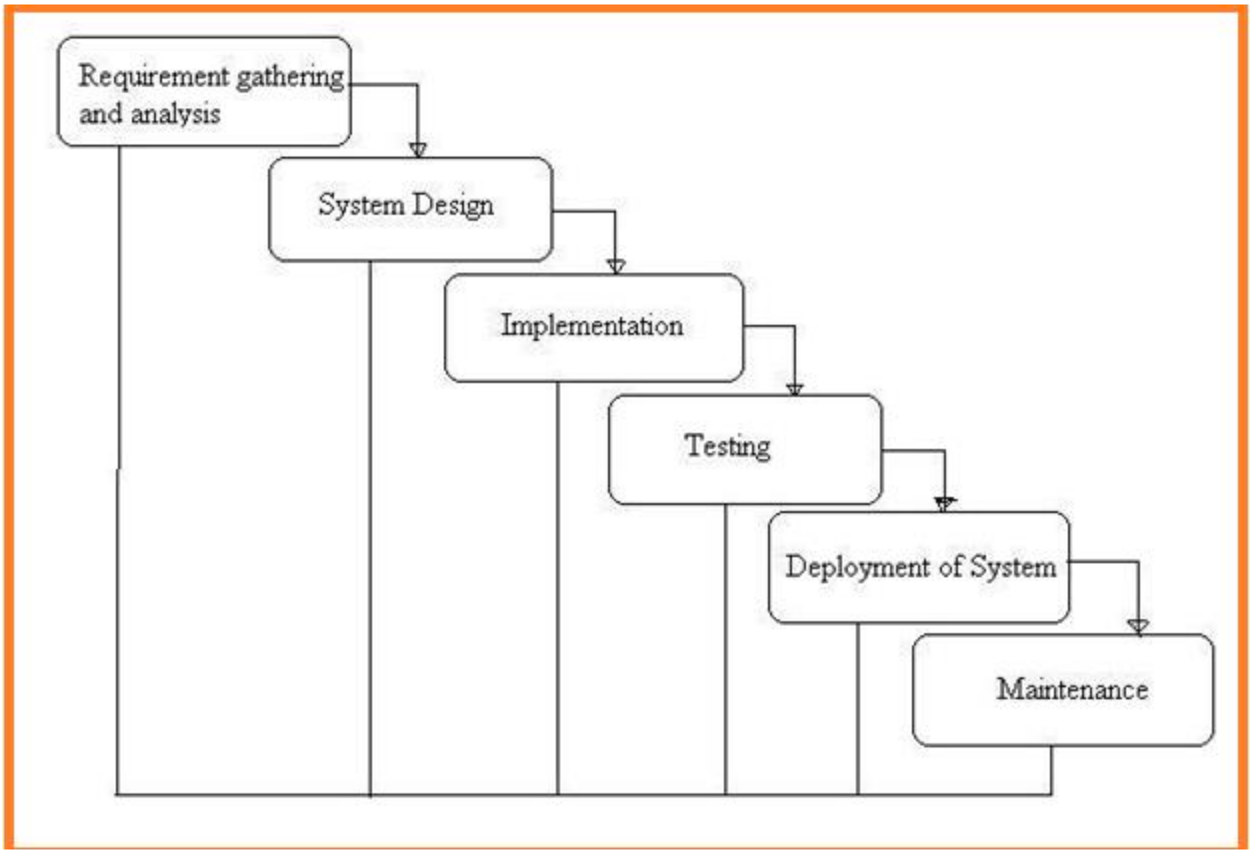
### Software Requirements

| | |
|---|---|
| Technology | Python 3.6 |
| Operating System | Windows Family |
| IDE | VS Code |
| Technology | Python, Django |
| Database Server | Postgresql |
| Front Design Technology | HTML, CSS, JS |

## INTRODUCTION

## LITERATURE SURVEY

## SYSTEM ANALYSIS



Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Maintenance – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**FUNCTIONAL REQUIREMENTS**

- **Student Login**
- **Student Signup**
- **Guest Login**
- **Guest Signup**
- **Canteen In charge**
- **Add cat**
- **Add item**
- **Feedback**
- **Search**
- **Upload dataset**
- **View cart**

**NON-FUNCTIONAL REQUIREMENT**

(NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *"how fast* *does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

Examples of Non-functional requirements

Here, are some examples of non-functional requirement:

1. Users must change the initially assigned login password immediately after the first successful login. Moreover, the initial should never be reused.
2. Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
3. Every unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail.
4. A website should be capable enough to handle 20 million users with affecting its performance

5. The software should be portable. So moving from one OS to other OS does not create any problem.
6. Privacy of information, the export of restricted technologies, intellectual property rights, etc. should be audited.

Advantages of Non-Functional Requirement

Benefits/pros of Non-functional testing are:

- The nonfunctional requirements ensure the software system follow legal and compliance rules.
- They ensure the reliability, availability, and performance of the software system
- They ensure good user experience and ease of operating the software.
- They help in formulating security policy of the software system.

Disadvantages of Non-functional requirement

Cons/drawbacks of Non-function requirement are:

- None functional requirement may affect the various high-level software subsystem
- They require special consideration during the software architecture/high-level design phase which increases costs.
- Their implementation does not usually map to the specific software sub-system,
- It is tough to modify non-functional once you pass the architecture phase.

KEY LEARNING

- A non-functional requirement defines the performance attribute of a software system.
- Types of Non-functional requirement are Scalability Capacity, Availability, Reliability, Recoverability, Data Integrity, etc.
- Example of Non-Functional Requirement is Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
- Functional Requirement is a verb while Non-Functional Requirement is an attribute
- The advantage of Non-functional requirement is that it helps you to ensure good user experience and ease of operating the software
- The biggest disadvantage of Non-functional requirement is that it may affect the various high-levelsoftwaresubsystems.

## Methods

### SOFTWARE OVERVIEW

#### History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).  Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### Input as CSV File

Reading data from CSV(comma separated values) is a fundamental necessity in Data Science. Often, we get data from various sources which can get exported to CSV format so that they can be used by other systems. The Panadas library provides features using which we can read the CSV file in full as well as in parts for only a selected group of columns and rows.

The CSV file is a text file in which the values in the columns are separated by a comma. Let's consider the following data present in the file named input.csv. You can create this file using windows notepad by copying and pasting this data. Save the file as input.csv using the save As All files(*.*) option in notepad.

```
import pandas as pd

data = pd.read_csv('path/input.csv')

print (data)
```

#### Operations using NumPy

NumPy is a Python package which stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Using NumPy, a developer can perform the following operations

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

## Django Web framework for Python

### Introduction to Django

If you go to the Web site djangoproject.com using your Web browser — or, depending on the decade in which you're reading this destined-to-be-timeless literary work, using your cell phone, electronic notebook, shoe, or any Internet-superceding contraption — you'll find this explanation: "Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design." That's a mouthful — or eyeful or pixelful, depending on whether this book is being recited, read on paper or projected to you on a Jumbotron, respectively. Let's break it down. Django is a high-level Python Web framework… A high-level Web framework is software that eases the pain of building dynamic Web sites. It abstracts common problems of Web development and provides shortcuts for frequent programming tasks. For clarity, a dynamic Web site is one in which pages aren't simply HTML documents sitting on a server's filesystem somewhere. In a dynamic Web site, rather, each page is generated by a computer program — a so-called "Web application" — that you, the Web developer, create. A Web application may, for instance, retrieve records from a database or take some action based on user input. A good Web framework addresses these common concerns:

● It provides a method of mapping requested URLs to code that handles requests. In other words, it gives you a way of designating which code should execute for which URL. For instance, you could tell the framework, "For URLs that look like /users/joe/, execute code that displays the profile for the user with that username."

● It makes it easy to display, validate and redisplay HTML forms. HTML forms are the primary way of getting input data from Web users, so a Web framework had better make it easy to display them and handle the tedious code of form display and redisplay (with errors highlighted). ● It converts user-submitted input into data structures that can be manipulated conveniently. For example, the framework could convert HTML form submissions into native data types of the programming language you're using.

● It helps separate content from presentation via a template system, so you can change your site's look-and-feel without affecting your content, and vice-versa.

● It conveniently integrates with storage layers — such as databases — but doesn't strictly require the use of a database.

● It lets you work more productively, at a higher level of abstraction, than if you were coding against, say, HTTP. But it doesn't restrict you from going "down" one level of abstraction when needed.

● It gets out of your way, neglecting to leave dirty stains on your application such as URLs that contain ".aspx" or ".php".

## DATA BASE DESIGN

### PostgreSQL

PostgreSQL, also known as Postgres, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and technical standards compliance. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users. It is the default database for macOS Server, and is also available for Linux, FreeBSD, OpenBSD, and Windows. PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. PostgreSQL is developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors.

**Data types**

A wide variety of native data types are supported, including:

Boolean

Arbitrary precision numerics

Character (text, varchar, char)

Binary

Date/time (timestamp/time with/without timezone, date, interval)

Money

Enum

Bit strings

Text search type

Composite

HStore, an extension enabled key-value store within PostgreSQL

Arrays (variable length and can be of any data type, including text and composite types) up to 1 GB in total storage size

Geometric primitives

IPv4 and IPv6 addresses

Classless Inter-Domain Routing (CIDR) blocks and MAC addresses

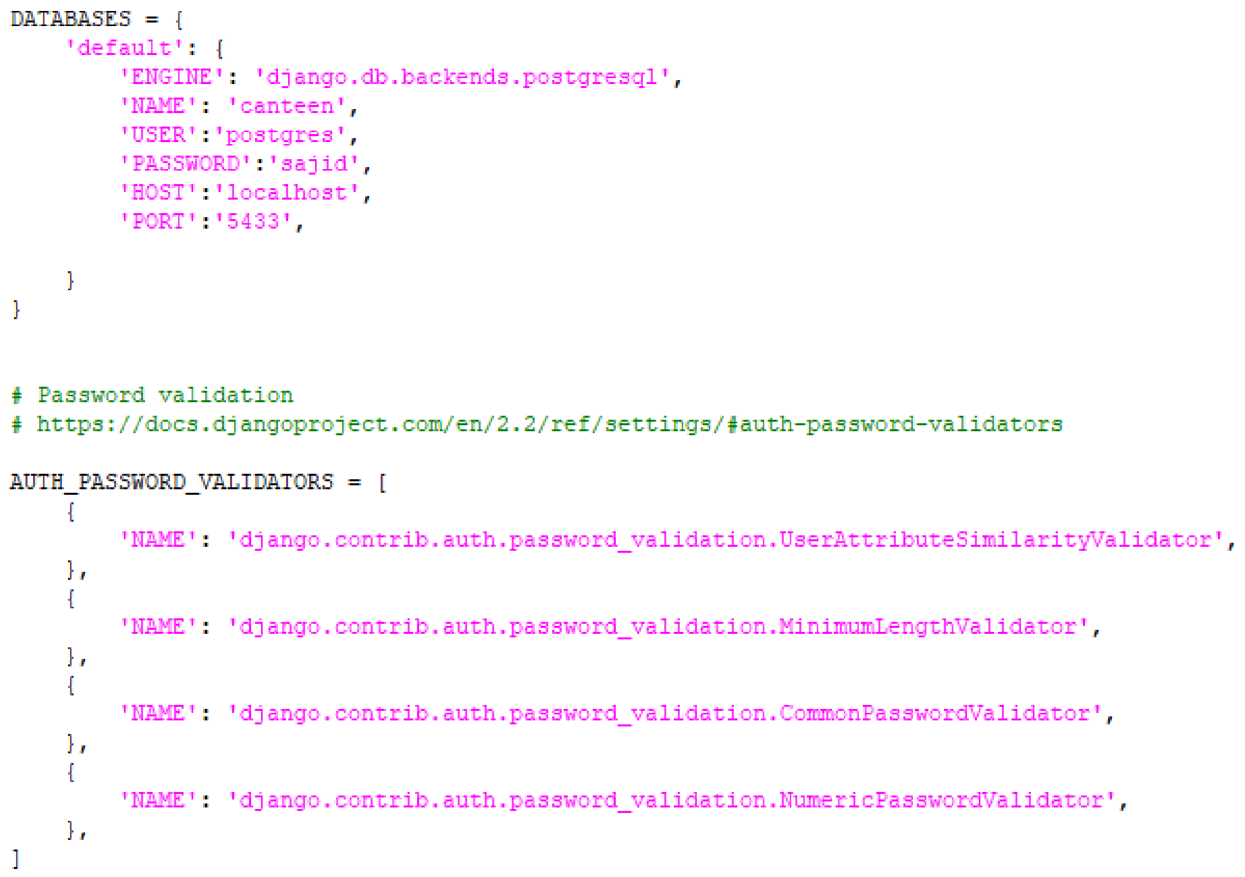
XML supporting XPath queries

Universally unique identifier (UUID)

**DATA BASE TABLES**

```
class student(models.Model):
        name=models.CharField(max_length=100);
        email=models.CharField(max_length=100);
        pwd=models.CharField(max_length=100);
        zip=models.CharField(max_length=100);
        gender=models.CharField(max_length=100);
        age=models.CharField(max_length=100);
class guest(models.Model):
        name=models.CharField(max_length=100);
        email=models.CharField(max_length=100);
        pwd=models.CharField(max_length=100);
        zip=models.CharField(max_length=100);
        gender=models.CharField(max_length=100);
        age=models.CharField(max_length=100);
class category(models.Model):
        catname=models.CharField(max_length=100);
class items(models.Model):
        id = models.AutoField(primary_key=True);
        catname=models.CharField(max_length=100);
        itemname=models.CharField(max_length=100);
        itemtype=models.CharField(max_length=100);
        itemcost=models.CharField(max_length=100);
        photo = models.CharField(max_length=100);
        ratingval=models.CharField(max_length=100);
class cart(models.Model):
        sno = models.AutoField(primary_key=True);
        sname=models.CharField(max_length=100);
        semail=models.CharField(max_length=100);
        itemname=models.CharField(max_length=100);
        itemcost=models.CharField(max_length=100);
        tot = models.CharField(max_length=100);
        totcost = models.CharField(max_length=100);
        status = models.CharField(max_length=100);
        otp = models.CharField(max_length=100);
class otp(models.Model):
        otp = models.CharField(max_length=100);
        decision = models.CharField(max_length=100);
        delivery = models.CharField(max_length=100);
        dat_e=models.CharField(max_length=100);
class sales(models.Model):
        year = models.CharField(max_length=100);
        month = models.CharField(max_length=100);
        yearmonth = models.CharField(primary_key=True,max_length=100);
        tamt = models.CharField(max_length=100);
class productsales(models.Model):
        year = models.CharField(max_length=100);
        month = models.CharField(max_length=100);
        yearmonth = models.CharField(max_length=100);
        tamt = models.CharField(max_length=100);
        prod = models.CharField(max_length=100);
class feedback(models.Model):
        pid = models.CharField(max_length=100);
        review = models.CharField(max_length=1000);
        ratingval=models.CharField(max_length=100);
```

**DATA BASE CONNECTION**



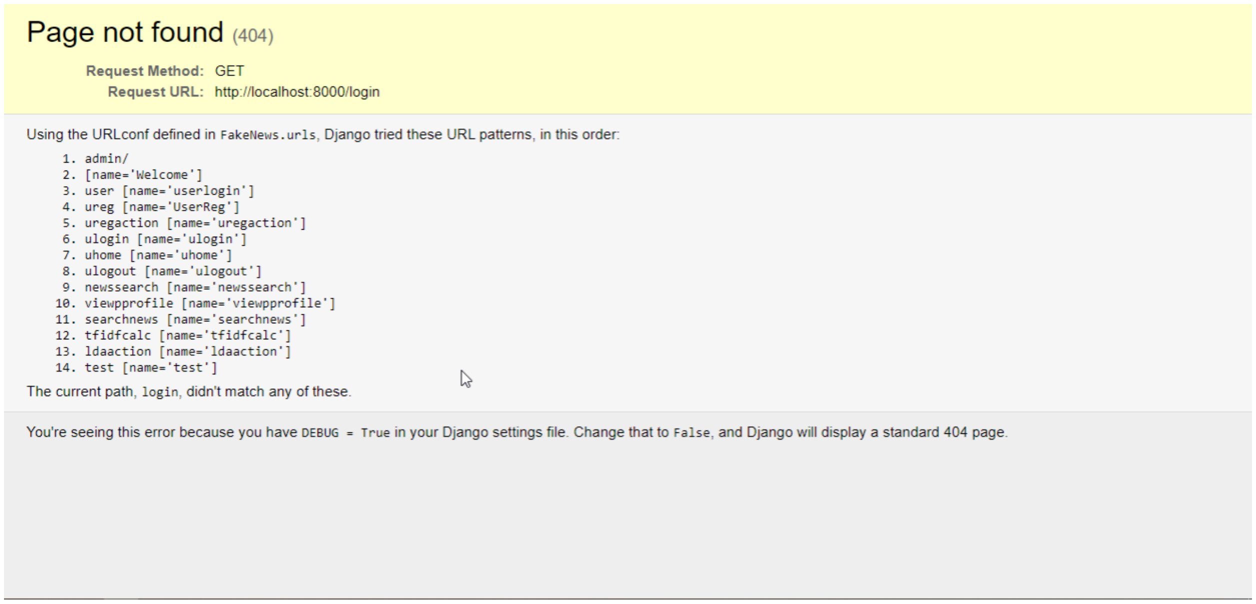
```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'canteen',
        'USER':'postgres',
        'PASSWORD':'sajid',
        'HOST':'localhost',
        'PORT':'5433',

    }
}


# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

## SOFTWARE TESTING

### Unit testing

### URL Mismatch Error

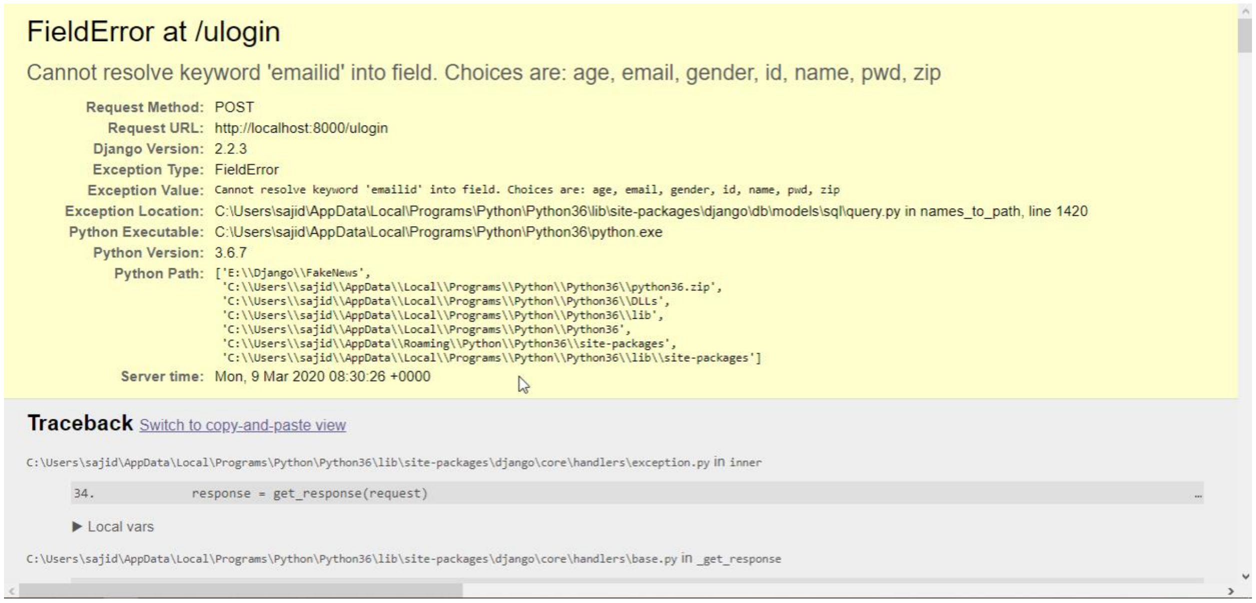
When we give URL like localhost:8000/login and if it's not match in urls.py files, we can get this error.



```
Page not found (404)
    Request Method:  GET
       Request URL:  http://localhost:8000/login

Using the URLconf defined in FakeNews.urls, Django tried these URL patterns, in this order:
    1. admin/
    2. [name='Welcome']
    3. user [name='userlogin']
    4. ureg [name='UserReg']
    5. uregaction [name='uregaction']
    6. ulogin [name='ulogin']
    7. uhome [name='uhome']
    8. ulogout [name='ulogout']
    9. newssearch [name='newssearch']
   10. viewpprofile [name='viewpprofile']
   11. searchnews [name='searchnews']
   12. tfidfcalc [name='tfidfcalc']
   13. ldaaction [name='ldaaction']
   14. test [name='test']
The current path, login, didn't match any of these.

You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.
```

### FieldError

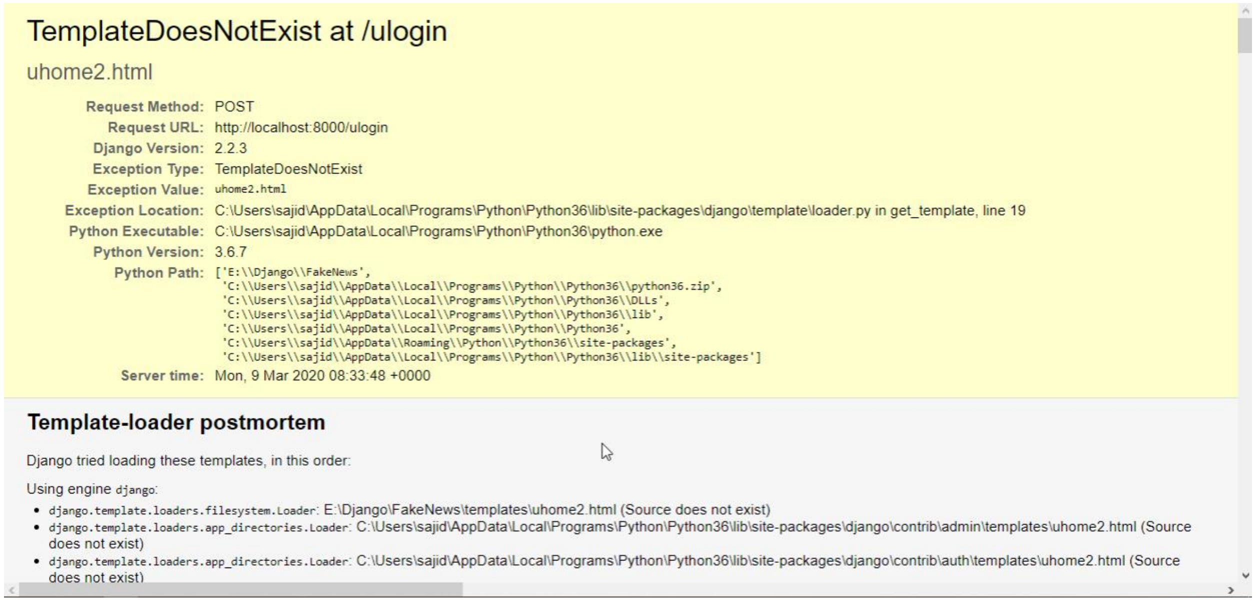
Database field mismatch from model. Given keyword 'emailid' into field. expected: age, email, gender, id, name, pwd, zip.



```
FieldError at /ulogin
Cannot resolve keyword 'emailid' into field. Choices are: age, email, gender, id, name, pwd, zip

      Request Method:  POST
         Request URL:  http://localhost:8000/ulogin
      Django Version:  2.2.3
      Exception Type:  FieldError
     Exception Value:  Cannot resolve keyword 'emailid' into field. Choices are: age, email, gender, id, name, pwd, zip
  Exception Location:  C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\db\models\sql\query.py in names_to_path, line 1420
   Python Executable:  C:\Users\sajid\AppData\Local\Programs\Python\Python36\python.exe
      Python Version:  3.6.7
         Python Path:  ['E:\\Django\\FakeNews',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36\\python36.zip',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36\\DLLs',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36\\lib',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36',
                        'C:\\Users\\sajid\\AppData\\Roaming\\Python\\Python36\\site-packages',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36\\lib\\site-packages']
         Server time:  Mon, 9 Mar 2020 08:30:26 +0000

Traceback Switch to copy-and-paste view

C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\core\handlers\exception.py in inner

    34.             response = get_response(request)                                             …

    ▶ Local vars

C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\core\handlers\base.py in _get_response
```

### TemplateDoesNotExist

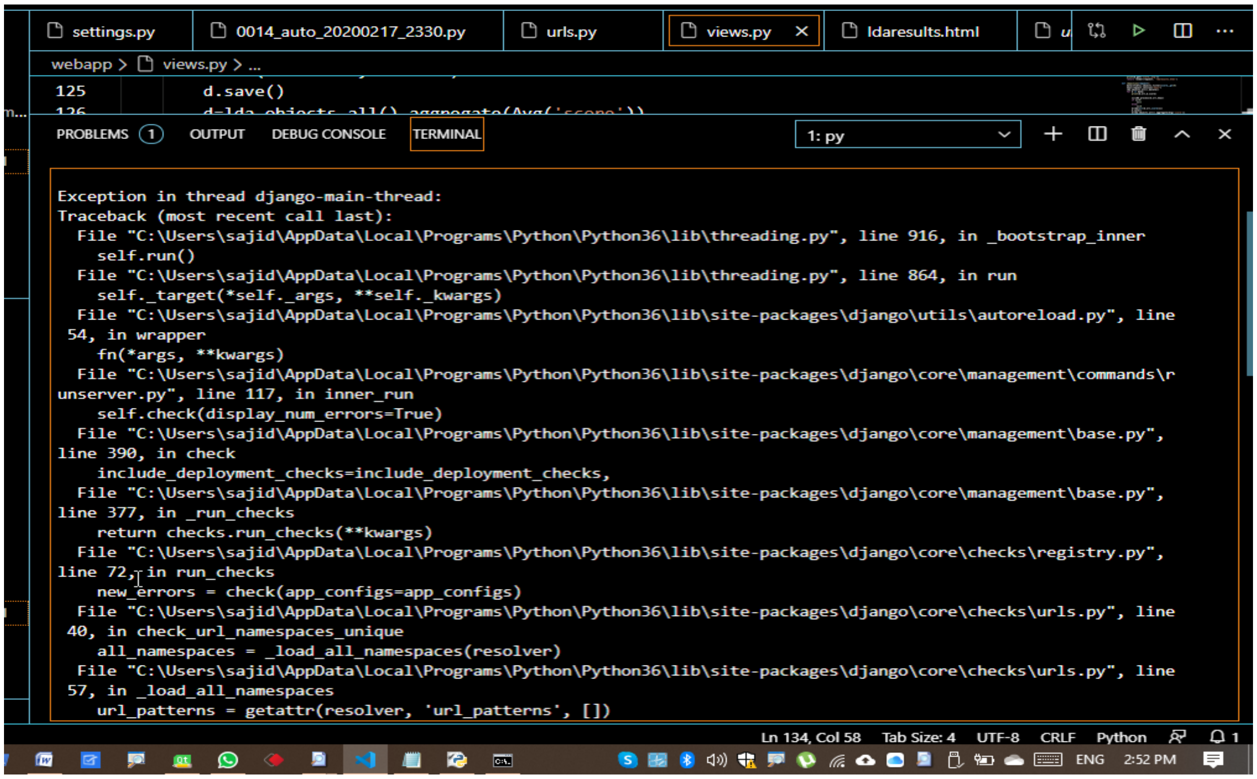
We get this error when URL redirecting and template file (html) not found, or mismatch



```
TemplateDoesNotExist at /ulogin
uhome2.html

      Request Method:  POST
         Request URL:  http://localhost:8000/ulogin
      Django Version:  2.2.3
      Exception Type:  TemplateDoesNotExist
     Exception Value:  uhome2.html
  Exception Location:  C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\template\loader.py in get_template, line 19
   Python Executable:  C:\Users\sajid\AppData\Local\Programs\Python\Python36\python.exe
      Python Version:  3.6.7
         Python Path:  ['E:\\Django\\FakeNews',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36\\python36.zip',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36\\DLLs',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36\\lib',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36',
                        'C:\\Users\\sajid\\AppData\\Roaming\\Python\\Python36\\site-packages',
                        'C:\\Users\\sajid\\AppData\\Local\\Programs\\Python\\Python36\\lib\\site-packages']
         Server time:  Mon, 9 Mar 2020 08:33:48 +0000

Template-loader postmortem

Django tried loading these templates, in this order:

Using engine django:
 • django.template.loaders.filesystem.Loader: E:\Django\FakeNews\templates\uhome2.html (Source does not exist)
 • django.template.loaders.app_directories.Loader: C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\contrib\admin\templates\uhome2.html (Source
   does not exist)
 • django.template.loaders.app_directories.Loader: C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\contrib\auth\templates\uhome2.html (Source
   does not exist)
```

### Exception while running server

If any syntactically mistakes in views.py we get this exception.



```
            d.save()
```

```
Exception in thread django-main-thread:
Traceback (most recent call last):
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\threading.py", line 916, in _bootstrap_inner
    self.run()
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\threading.py", line 864, in run
    self._target(*self._args, **self._kwargs)
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\utils\autoreload.py", line
 54, in wrapper
    fn(*args, **kwargs)
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\core\management\commands\r
unserver.py", line 117, in inner_run
    self.check(display_num_errors=True)
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\core\management\base.py",
line 390, in check
    include_deployment_checks=include_deployment_checks,
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\core\management\base.py",
line 377, in _run_checks
    return checks.run_checks(**kwargs)
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\core\checks\registry.py",
line 72, in run_checks
    new_errors = check(app_configs=app_configs)
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\core\checks\urls.py", line
 40, in check_url_namespaces_unique
    all_namespaces = _load_all_namespaces(resolver)
  File "C:\Users\sajid\AppData\Local\Programs\Python\Python36\lib\site-packages\django\core\checks\urls.py", line
 57, in _load_all_namespaces
    url_patterns = getattr(resolver, 'url_patterns', [])
```

## MANUAL TESTING ON PROJECT APPLICATION

### TEST CASES

| | | | | |
|---|---|---|---|---|
| **Test Case ID** #1 | | **Test Case Description** - Validations in Registration Form | | |
| **S#** | **Prerequisites** | **S#** | **Test Data Requirement** | |
| 1 | User should be Registered | 1 | Data should be valid | |
| **Test Condition** | | | | |
| Entering data in registration form | | | | |
| **Step #** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail/Not Executed/Suspended** |
| 1 | User gives First and Last Name | Pop showing email verification message | Enter valid email/password | Fail |
| 2 | Submitting the form without entering any details | Pop showing email verification message | Enter email /password | Fail |
| 3 | User enters invalid format of email id | Pop showing email verification message | Enter valid email id | Fail |
| 4 | User enters a phone number with < 10 digits | Pop showing email verification message | Enter valid phone number | Fail |
| 5 | Entering valid username and password | Pop showing email | Pop showing email verification message | Pass |

| | | | | |
|---|---|---|---|---|
| | | verification message | | |

Table 1 Registration test case

| Test Case ID #2 | | Test Case Description - Validations in Login Form | | |
|---|---|---|---|---|
| **S#** | **Prerequisites** | **S#** | **Test Data Requirement** | |
| 1 | User should have an email id | 1 | Data should be valid | |
| **Test Condition** | | | | |
| Entering data in login form | | | | |
| **Step #** | **Step Details** | **Expected Results** | **Actual Results** | **Pass/Fail/Not Executed/Suspended** |
| 1 | User gives aemail or password of <6 characters | User logged in | Enter valid email/password | Fail |
| 2 | Submitting the form without entering any details | User logged in | Enter email /password | Fail |
| 3 | User enters wrong Email and (or) password | User logged in | Enter correct email /password | Fail |

Table 2 Login test case

## CONCLUSION

The Capstone Project develops a web-based data solution application based on pre-set requirements and resolves an existing problem of having to manual system of placing order using an web application which will save the time and used machine learning algorithms to help canteen admin to predict the sales to increase the business. As this is a team Project, we have developed a real time experience working in  an IT project