PICKTICK

3907, Block A, Sushant Lok Phase I, Sector 28, Gurugram, Haryana 122022

Summer Internship Report

Employee Daily Progress Report

Kavin Sharma
B.tech 4th Year
CSE - OIL & GAS INFORMATICS
UPES

Date: 29th June, 2017

TABLE OF CONTENTS

TABLE OF CONTENTS

1 ACKNOWLEDGEMENTS **2**

2 INTRODUCTION AND OVERVIEW **3-4**

2.1 Picktick India – Background Information **3**

2.2 Project Background and Scope **3**

2.3 Direction of Research at Daily Employee Progress Report Web App **4**

2.4 Project Overview **4**
2.7.1 Part-I: Employee Login Panel 4
2.7.2 Part-II: Admin Login Panel 4

3 PROJECT PART I: EMPLOYEE LOGIN PANEL 6 -11

3.1 Libraries Used **5**

3.2 Main Function **6**
3.3.1 SignIn Email / Password 7
3.3.2 Home Template 7
3.3.3 Signup 8
3.3.4 Create Report 9
3.3.5 Check Report 10
3.3.6 Full Deatail of a Report 11
3.3.7 Lost Password 11

4 PROJECT – II: ADMIN LOGIN PANEL **12-16**

4.1 Libraries Used **12**

4.2 Main Function **14**
4.3.1 Checking Submitted Reports and Account Status 14
4.3.2 Search for all reports for a specific Employee 15
4.3.3 Full Detail for a Single Day Report 16

1 Acknowledgements

Firstly, I would like to express my sincere gratitude to my mentor Mr. Nishant Thapliyal for the continuous support in this internship project and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and give this unique Daily Employee Progress Report Web App in my final submission.

I could not have imagined having a better advisor and mentor for this internship period at Picktick India.

Besides my mentor I would like to thank the rest of Picktick India Team: Ms. Amrita Kashyap, and Mr. Nikhil Ahuja, for their help in testing, review and continuous Tech support. It was great help of Mr Rahul Jain for giving the overview of whole report creation process.
My sincere thanks also goes to Mr Sahil Bhutani who provided me an opportunity to join as intern.

2 Introduction and Overview

2.1 Picktick India – Background Information

An Augmented Reality based City Discovery Platform.
Picktick is a platform to discover 1000+ things to do near using augmented reality. Dive into a whole new world with Augmented Reality and explore your city from monuments to deals to restaurants to shopping to marathons all in one place

An Event Aggregator Platform

2.2 Project Background and Scope

Do you enjoy doing paper work rather than being on the job site? We definitely don't! But daily reports are important to know if a project is ahead or behind schedule. That's why we created Python Django based Daily employee Progress Report Web App.

2.2.1 Daily Employee Progress Report perspective

In many offices and businesses, daily task reporting is a norm. This practice helps companies keep track of the work accomplished by employees on projects which are ongoing. As a consequence, based on the reporting, the higher authorities are in the know about developments, overall progress etc. Whether you're a manager or supervisor, your reporting authority is going to expect daily reports from you. Let our daily report web-app make the job simpler for you.

Features:

- Employee
  - Employee Login/Signup.
  - Password Reset Option.
  - Create / Check Report Functionality.
  - Reports Sorted according to Date/Time from latest to previous reports.
  - Reports Search functionality.

- Admin Panel
  - Admin Login.
  - Employee Account Enable /Disable functionality.
  - Check Report Functionality.
  - Reports Sorted according to Date/Time from latest to previous reports.
  - Reports Search functionality.

2.3 Direction of Research at Daily Employee Progress Report Web App.

Created project includes Two Panel Employee | Admin Panel. Employee can create new reports and after creating reports both employee and admin can check reports which are sorted according to date and time additionally web application includes search report functionally in which reports can be searched out using words included in task and at last includes Account Logout functionality included in both admin and employee panel .
Admin panel also includes the functionality of Employee Account Disable functionality.

2.4 Project Overview

2.4.1 Part-I: Employee Login Panel

This Part of project includes Login of Employee for creating daily progress report of task assigned to employee. Feature of Employee Panel includes:

- Create Report
- Check Report
- Search Reports by Task

2.4.2 Part-II: Admin Login Panel

This Part of project includes Admin Login for web app by which admin can manage employees and check the progress of task assingned to employee. Features of Admin panel includes:

- Select Employee by username
- Enable/Disable Employee Account
- Check Report
- Search Reports by Task

3 Project Part - I: Employee Login Panel

3.1 Libraries Used:

- import datetime

  Intro: The datetime module supplies classes for manipulating dates and times in both simple and complex ways.More

  Installation: pre installed in python 2.7 and above

  Usage: used for converting start datatime and end datetime to seconds

```
Stdatetime = "21 july 2017 11:00"
dt_obj = time.strptime(str(stdatetime), "%d %B %Y %H:%M")
    timestamp = time.mktime(dt_obj)
```

Getting datetime in seconds

- from django.http import HttpResponse

  Intro: Django function used for showing response on each request.

  Installation: Predefined in Django views.py

  Usage: Printing Scraped Event Headings on WebPage.

List Which Includes Each Event Title

```
return HttpResponse(fetch)
```

- from firebase import firebase

  Intro: Python interface to the Firebase's REST API more

  Installation: pip3 install python-firebase

  Usage: used for performing CURD in a database and Firebase Authentication

- import pyrebase

  Intro: A simple python wrapper for the Firebase API.more

  Installation: pip3 install pyrebase

  Usage: used for updating Task and Progress to firebase and for authentication of employee

- from django.contrib import auth

Intro: Using the Django authentication system.more

Installation: preinstalled in django

Usage: logout the user

- Import pytz

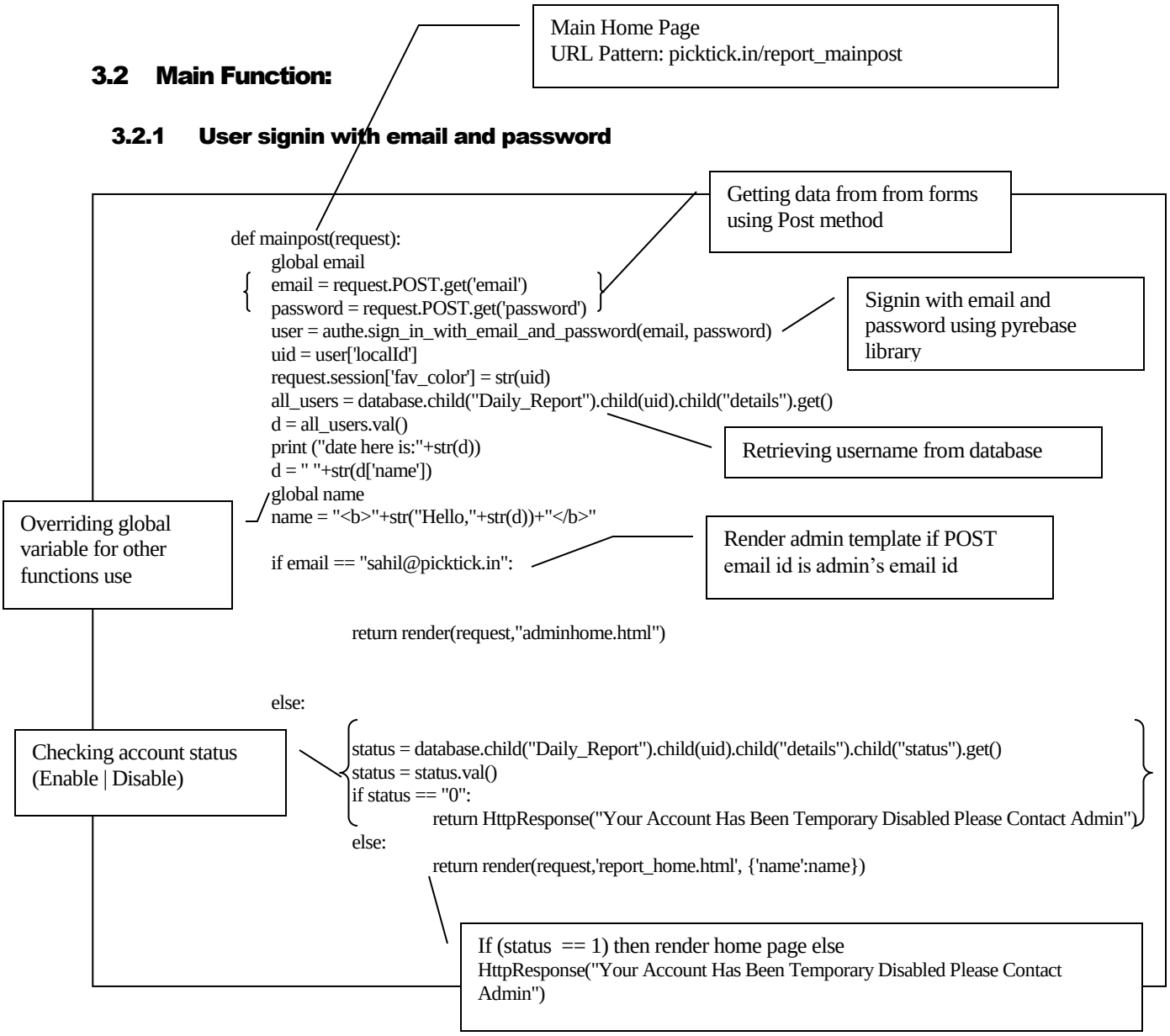
Intro: Library support for time zones.more

Installation: preinstalled in django

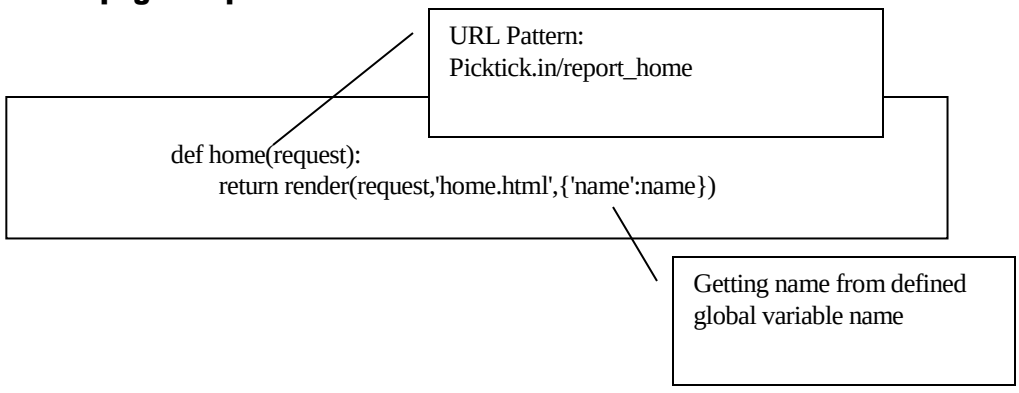
Usage: specifying UTC +5:30 timezone while fetching system time

3.2 Main Function:

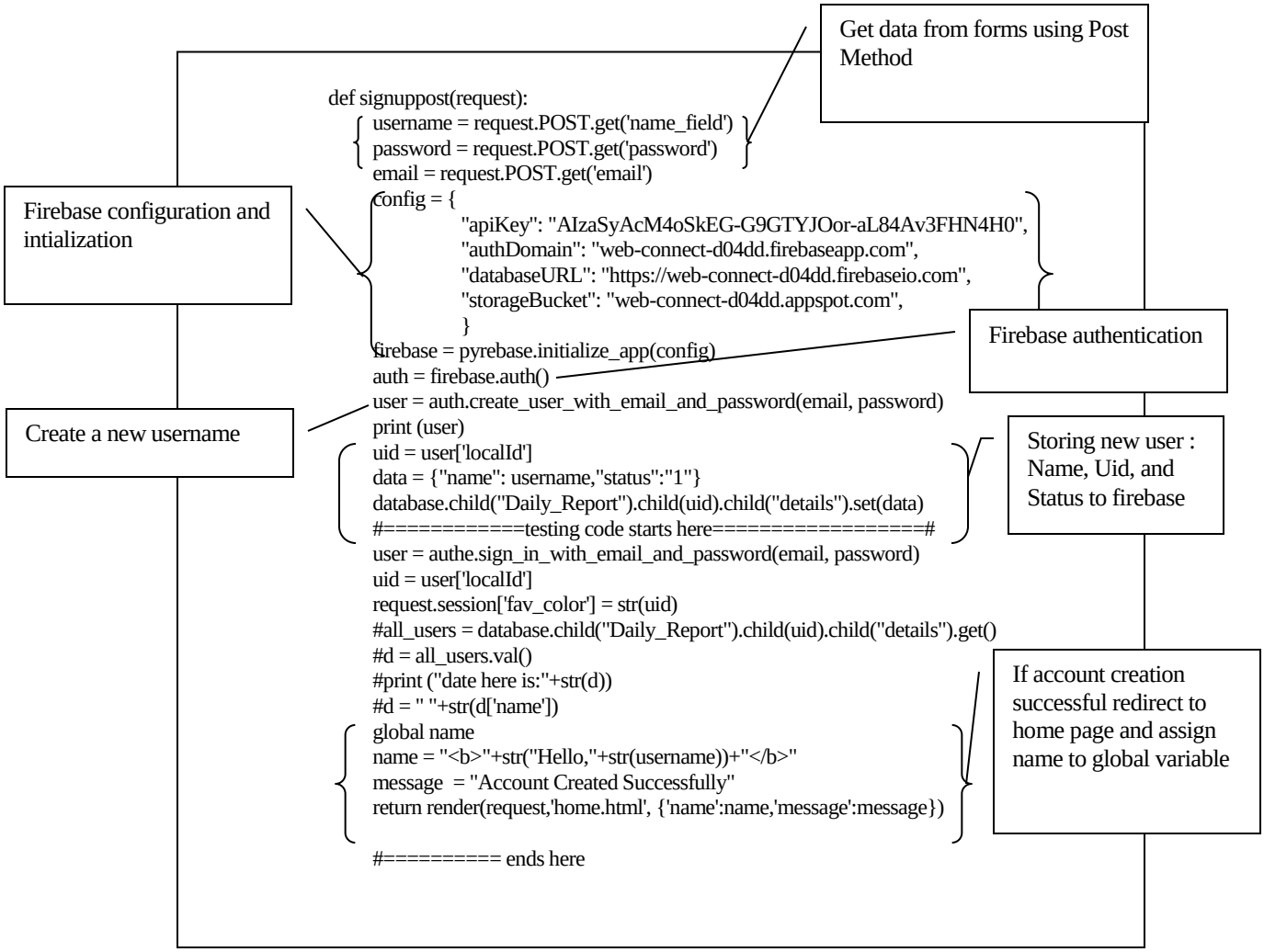
3.2.1 User signin with email and password



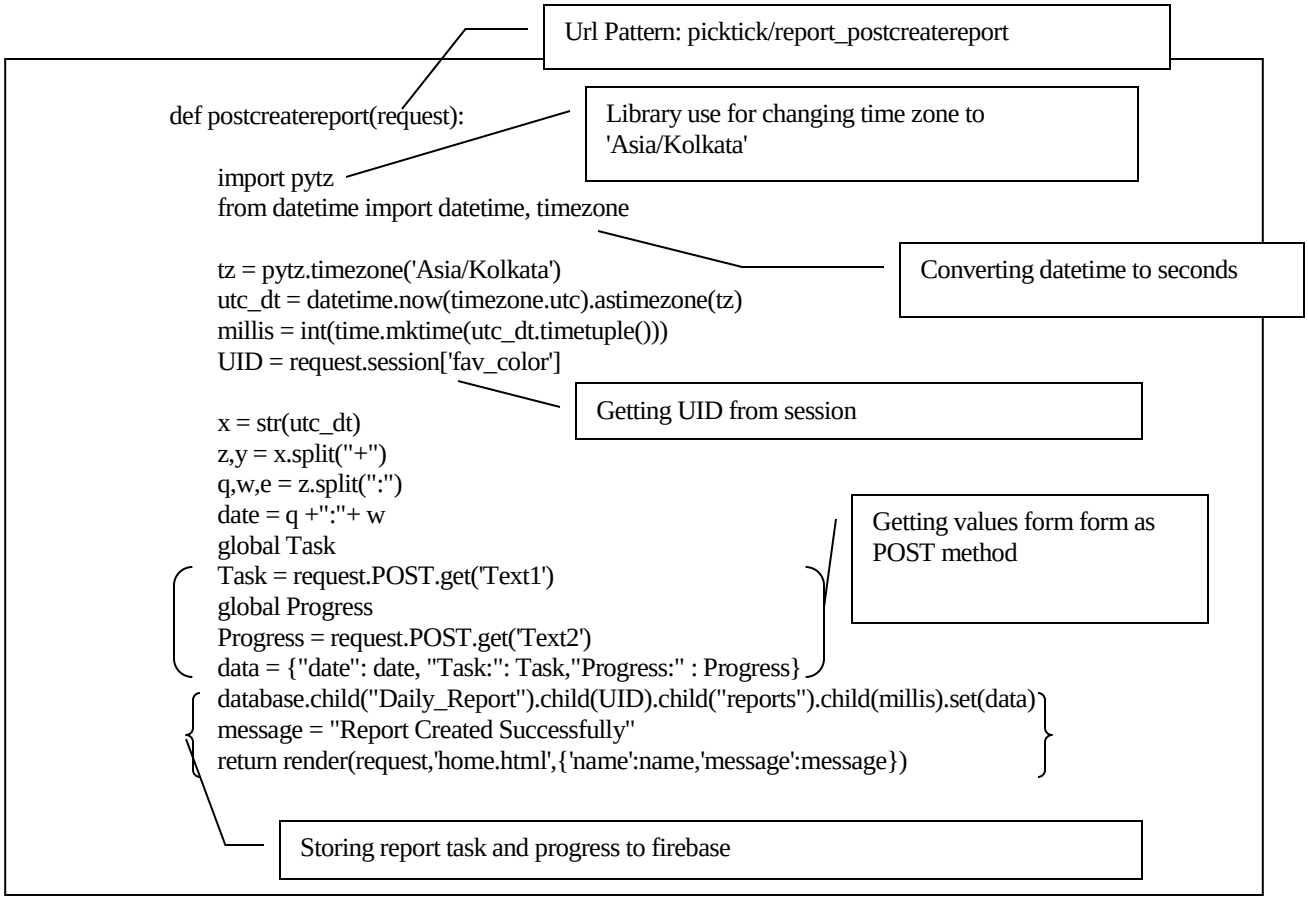
3.2.2 Homepage template for Back Button



3.2.3 Signup or Create New User Account with email and password



3.2.4 Create New Report and storing data to firebase



3.2.5 Check for Submitted Reports

URL Pattern: picktick.in/report_checkreport

```
def checkreport(request):
    import datetime
    UID = request.session['fav_color']
    all_user_ids = database.child("Daily_Report").child(UID).shallow().child('reports').get().val()
    if all_user_ids is None:
        return render(request, "noreport.html", {"name":name})
    lis = []
    for item in all_user_ids:
        lis.append(str(item))
        lis.sort(reverse = True)
#   print (lis)
    dat = []
    for i in lis:
        i = float(i)
        #i = (int(i)/1000.0)
#       print ("in seconds"+str(i))
        a = datetime.datetime.fromtimestamp(i).strftime('%H:%M %d-%m-%Y')
#       print ("in time "+ str(a))
        dat.append(str(a))
#   print (dat)
    task = []
    for title in lis:
        b = database.child("Daily_Report").child(UID).child('reports').shallow().child(title).child('Task:').get().val()
        c = b.split(" ")
        print (len(c))
        if len(c)<10:
            task.append(b)
        else :
            ta = " ".join(c[:10])
            ta = str(ta) + "..."
            task.append(ta)
    print(task)
    lise = zip(lis, dat, task)
    return render(request, "check.html", {"lise" :lise, "name":name, "UID":UID})
```

Requesting user UID from session

Extracting all timestamps from firebase

If no report found then render no report.html

Sort time stamps form latest to last

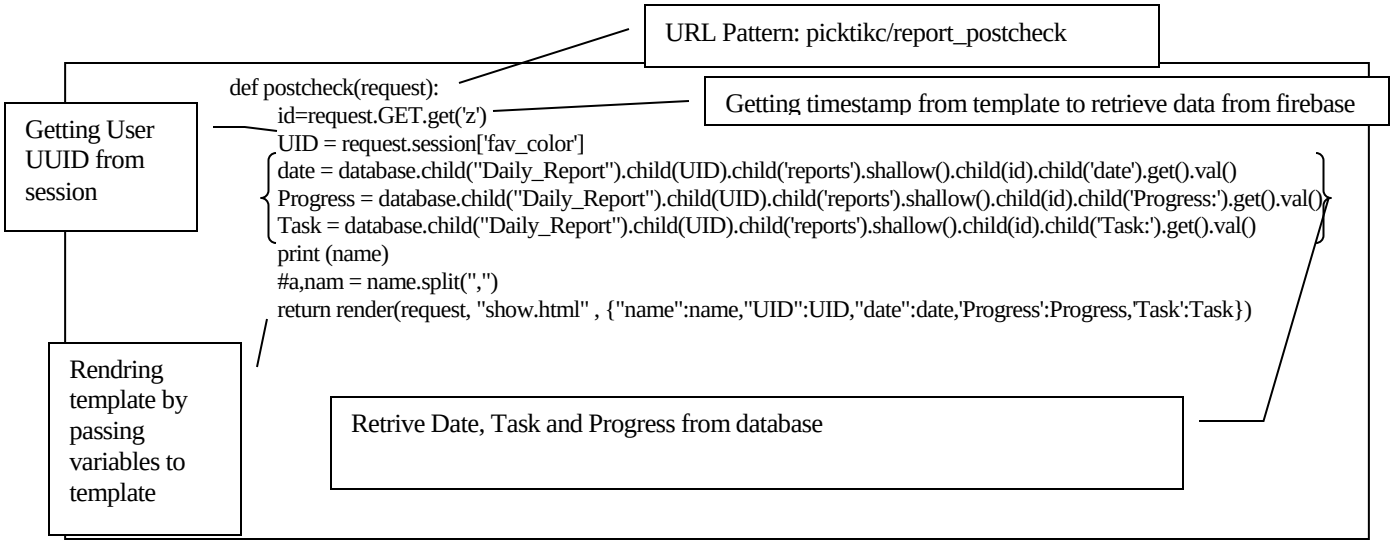
Convert timestamps to date and time

Retrieve task and progress from firebase in a sorted way to a list

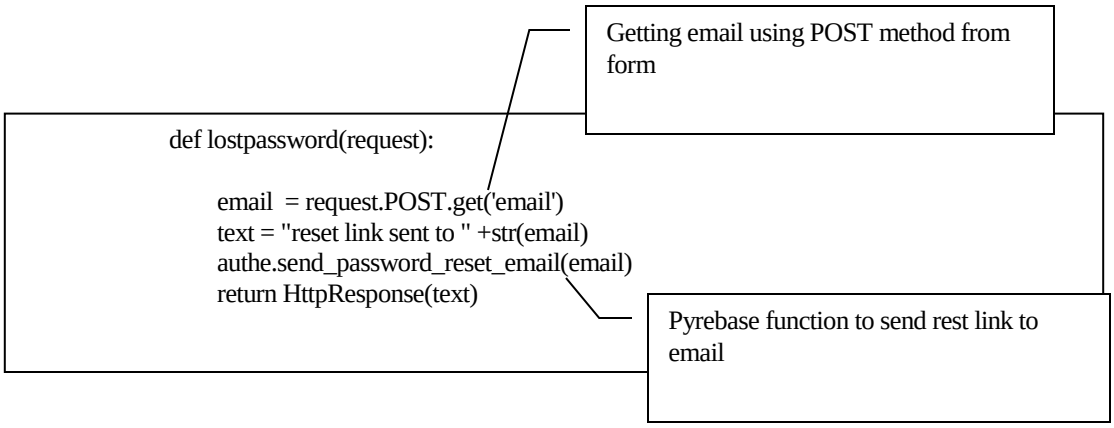
Zipping 3 lists to one list for running lists simultaneously

Rendring template and passing name, UID, lise as a variable to template

3.2.6 Full Detail for a Single Day Report



3.2.7 Recovering Lost Password



4 Project Part - II: Admin Login Panel

4.1 Libraries Used:

- import datetime

  Intro: The datetime module supplies classes for manipulating dates and times in both simple and complex ways.More

  Installation: pre installed in python 2.7 and above

  Usage: used for converting start datatime and end datetime to seconds

```
Stdatetime = "21 july 2017 11:00"
dt_obj = time.strptime(str(stdatetime), "%d %B %Y %H:%M")
    timestamp = time.mktime(dt_obj)
```

Getting datetime in seconds

- from django.http import HttpResponse

  Intro: Django function used for showing response on each request.

  Installation: Predefined in Django views.py

  Usage: Printing Scraped Event Headings on WebPage.

List Which Includes Each Event Title

```
return HttpResponse(fetch)
```

- from firebase import firebase

  Intro: Python interface to the Firebase's REST API more

  Installation: pip3 install python-firebase

  Usage: used for performing CURD in a database and Firebase Authentication

- import pyrebase

  Intro: A simple python wrapper for the Firebase API.more

  Installation: pip3 install pyrebase

  Usage: used for updating Task and Progress to firebase and for authentication of employee

- from django.contrib import auth

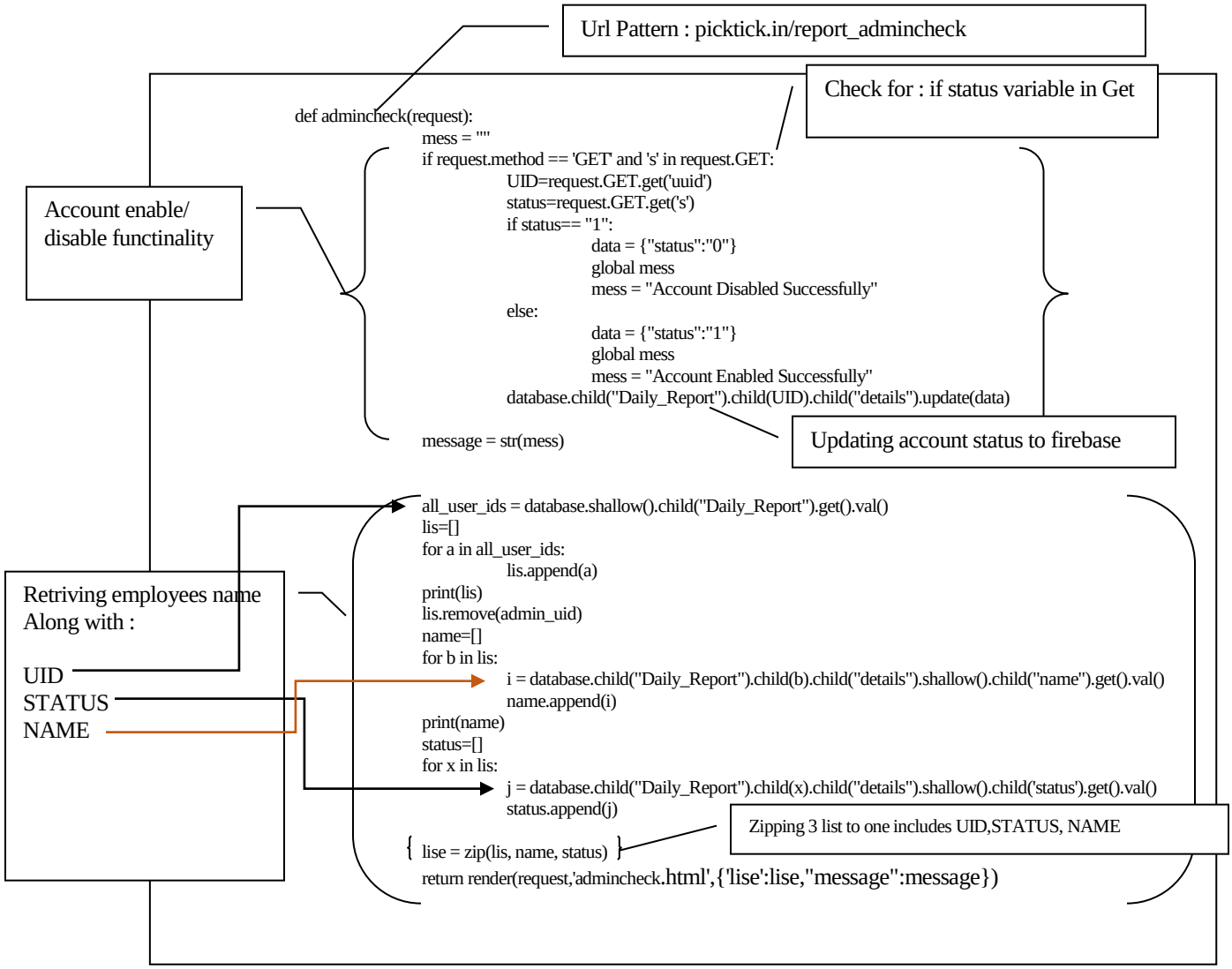
Intro: Using the Django authentication system.more

Installation: preinstalled in django

Usage: logout the user

4.2 Main Function:

4.2.1 Check for Submitted Reports and Account Status



4.2.2 Check and Search for all reports for a specific Employee

```
def postadmincheck(request):
    import datetime
    if request.method == 'GET' and 'csrfmiddlewaretoken' in request.GET:

        UID=request.GET.get('UUID')
        search = request.GET.get('search')
        search = str(search).lower()
        timestamp = database.child("Daily_Report").child(UID).shallow().child('reports').get().val()
        task = []
        for id in timestamp:
                tas = database.child("Daily_Report").child(UID).child('reports').shallow().child(id).child('Task:').get().val()
                tas = str(tas) + "$" + str(id)
                task.append(tas)
        matching = [str(string) for string in task if str(search) in (string.lower())]
        ftask=[]
        ftime=[]
        for d in matching:
                task,time = d.split("$")
                ftask.append(task)
                ftime.append(time)
        dat = []
        for date in ftime:
                i = float(date)
                a = datetime.datetime.fromtimestamp(i).strftime('%H:%M %d-%m-%Y')
                dat.append(str(a))

        name = database.child("Daily_Report").child(UID).child("details").shallow().child("name").get().val()

        lise = zip(ftask, ftime, dat)
        return render(request,"search.html",{"UID":UID

    else:
        import datetime
        q1=request.GET.get('q')
        ide = str(q1)
        all_user_ids = database.child("Daily_Report").child(ide).shallow().child('reports').get().val()
        print (all_user_ids)
        lis = []
        for item in all_user_ids:
                lis.append(str(item))
                lis.sort()
                lis.reverse()

        dat = []
        for i in lis:
                i = float(i)
                a = datetime.datetime.fromtimestamp(i).strftime('%H:%M %d-%m-%Y')
                dat.append(str(a))
        task = []
        for title in lis:
                b = database.child("Daily_Report").child(ide).child('reports').shallow().child(title).child('Task:').get().val()
                c = b.split(" ")
                print (len(c))
                if len(c)<10:
                        task.append(b)
                else :
                        ta = " ".join(c[:10])
                        ta = str(ta) + "..."
                        task.append(ta)
                print(task)
        name = database.child("Daily_Report").child(ide).child("details").shallow().child("name").get().val()
        lise = zip(lis, dat, task)
        return render(request,"adminshow.html", {'lise':lise,'name':name, 'ide':ide})
```

Check if csrf token of search bar present in get method

def postadmincheck(request):

Retring all timestamp for a uid

Two Cases :

1 Searching Reports by task

2 Retriving all tasks for a specific user

Checking if searched value present in User task's

Appending datetime to output of matching tasks

Rendring search template along with name, uid and lise as a varialbe

Using get method to retrieve user id

Retriving all timestamps for a specific user

Appending sorted timestapms to list

Converting timestamp to datetime

Retrieving task for a specific time stamp and stroing it to a list

Retrieving user name from firebase

4.2.3 Full Detail for a Single Day Report