



A PROJECT
(Based on BCA - 601)
ON
DATA MINING

Submitted

By

VICKY KUMAR
PRASHANT KUMAR TIWARI

(Bachelor of Computer Application)
6th SEMESTER

Under the Guidance of

PROFESSOR PRABHAT KUMAR

TEAM MEMBERS:-

VICKY KUMAR

UNIVERSITY ROLL NO.:-214151

REG.NO.:-20CCVGCS035/20

PRASHANT KUMAR TIWARI

UNIVERSITY ROLL NO.:-214150

REG.NO.:-20CCVGCS034/20

Introduction

A well documented problem faced by maintainers when understanding a software system is the lack of familiarity with it, combined with the lack of accurate documentation [11]. Several techniques and methods have been proposed in order to facilitate this process consuming activity [3], [7], [9].

The work presented in this paper is part of a wider research effort investigating the applicability and suitability of using data mining to facilitate program comprehension and maintenance [4], [13], [15],[15]. This effort aims at developing a methodology for semi automated program comprehension incorporating data mining. A fundamental underlying assumption is that the maintainer may have little or no knowledge of the program which is analysed. The work presented here aims to help maintainers to recognise parts of C++ code that have common characteristics, facilitating program understanding. This work focuses on extracting data from C++ code which are clustered in order to identify logical, behavioural and structural correlations amongst program components. C++ was selected as it is widely used but is more complicated to comprehend, compared to other programming languages, like COBOL. As an object oriented language, it can be analyzed in either a more detailed, technical level (member data and member functions analysis), or in a more abstract level (class analysis).

The objectives of this work are:

- i) to define the input model needed to extract data from C++ code and populate a database. This requires defining program entities and their attributes.
- ii) to propose a pre-processing method that extracts data from code using the input data model.
- iii) to assess the feasibility of the methodology in producing valid, useful and novel patterns and knowledge about a software system. The remaining of this paper is organised as follows. Section 2 reviews previous solutions in the domain of data mining for program comprehension. Section 3 outlines the proposed methodology for pre-processing C++ source code, the input data model and the steps of this methodology. Section 4 assesses the accuracy of the output of this method, analyses its results and outlines deductions from its application. Finally, conclusions and directions for future work are presented in section 5.

Background

Software maintenance is the most difficult stage in software lifecycle, often performed with limited understanding of the design and the overall structure of a system because of commercial pressures [11]. Fast, unplanned modifications, based on partial understanding of a system, give rise to increased code complexity and deteriorated modularity, thus resulting in 50%-90% of the maintainers' time to be spent on program comprehension [14]. Furthermore it is recognised that there are no explicit guidelines given a program understanding task, nor there are good criteria to decide how to represent knowledge derived by and used for it [2].

Data mining and its ability to deal with vast amounts of data, has been considered a suitable solution in assisting software maintenance often resulting in remarkable results [1], [6], [8], [10], [12], [17], [17]. Our approach similarly uses data mining to get insights into systems design and structure [4], [13], [15], [15].

The following paragraphs briefly review some of the most prominent solutions in the area of data mining for software maintenance and compare these to our approach.

- Using Clustering to Produce High-Level System Organisations of Code

This solution proposes a collection of algorithms which facilitate the automatic recovery of the modular structure of a software system from its source code [8]. It creates a hierarchical view of the organization of the system based mainly on the components and the relationships that exist in the source code.

First it represents the system modules and the module-level relationships as a module - dependency graph. Then it partitions this graph so that the high - level subsystem structure can be derived from the component level relationships extracted from the source code. Based on the concepts of cohesion and coherence three parameters are introduced: intra-connectivity, inter connectivity and modularisation quality

The basic goal of this modularisation technique is to automatically partition the components of a system into clusters (subsystems) so that the resultant organization concurrently minimises interconnectivity while maximising intra-connectivity. The underlying assumption is that a well-designed system is organised into cohesive clusters that are loosely interconnected. The main drawback of this solution is that as the number of files exceeds 20, calculation time is greatly increased.

□ A Software Evaluation Model Using Component Association Views

This solution proposes a model for the evaluation of the architectural design of a system based on the association between the components of the system [12]. It allows measurement of system modularity, as an indication of the design quality and its decomposition into subsystems. For this reason the following three association views of a system are generated:

- i) Control passing: It represents the correlation among the system components based on function invocation.
- ii) Data exchange: It epitomises the correlation among the system components based on aggregate data types (except integer, real, boolean and string) that are either passed as parameters between two functions or are referenced by a function.
- iv) Data sharing: It signifies the correlation among the system components based on sharing the global variables by the functions. In this approach the software system is modelled as an attributed relational graph with system entities as nodes and data-control-dependencies as edges. At this point, the application of data mining techniques, like association rules helps the decomposition of the graph into domains of entities based on the association property. The next step is to populate a database of these domains. This approach is based on the concept of the association between the components of a system. There are however other characteristics that play crucial role in grouping system components, such as the number of member data or functions in a class. These can be discovered by using other data mining techniques like clustering.

□ A Method for Legacy Systems Maintenance by Mining Data Extracted from Code

This approach used data mining to facilitate software maintenance and reliability assessment. It addressed C/C++ and COBOL legacy systems aiming at understanding low/medium level concepts and relationships involving components at the function, paragraph or even line of code level [4], [13], [16].

This approach consists of three distinct phases: a)

data extraction,

b) data mining application

c) result evaluation.

There were different challenges in each phase. These involved the definition of an appropriate data model which captures as much information about the code as possible, the construction of a database suitable for data mining, the selection and customization and application of data mining algorithms and the assessment of the outcomes by domain experts. The approach deals with both COBOL and C/C++ programs and varies according to the differences between these languages. For C programs, we used functions as entities, and attributes defined according to the use and types of parameters and variables, and the types of returned values. We

then applied clustering to identify sub-sets of source code that are grouped together according to custom-made similarity metrics.

For COBOL programs we used paragraphs as entities, and binary attributes depending on the presence of user-defined and language-defined identifiers. In this case we derived association rules in order to establish inter-group and intragroup relationships

Results represent the syntactic and semantic content of the source code. Code is represented by means of models or graphs, like variable relationship model, a variable-block relationship model or even models that convey a meaning similar to Data Flow Diagrams and flow charts. Programs are abstracted into groups containing interrelated entities and are grouped together. This solution addresses systems both at medium and at low level and confirms that data mining can produce structural views of source code thus facilitating legacy systems understanding. There were however issues that had to do with correlations across system components such as programs and files. This deficiency was dealt with by the methodology proposed in this paper.

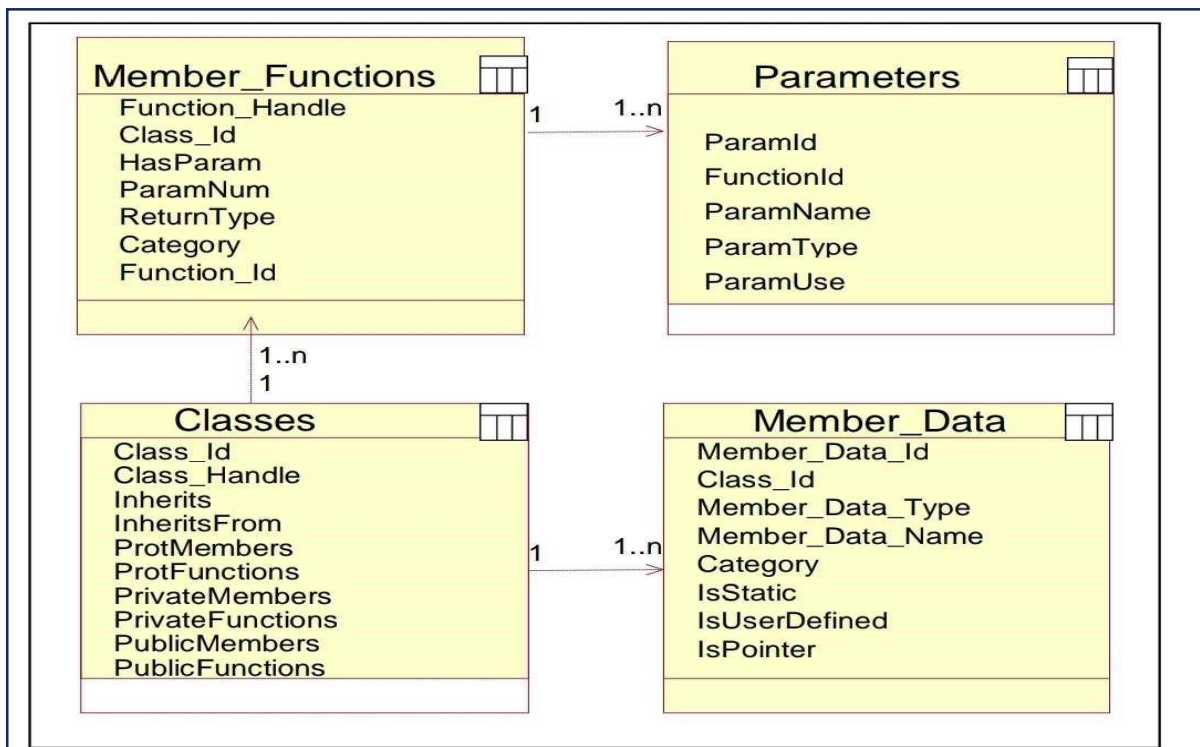
2. Description of the Proposed Framework

The framework proposed here, was developed for pre-processing C++ source code at the program level and consists of the following parts:

- i) The input model, which involves the specification of program entities and their attributes.
- ii) The pre-processing method.

This section outlines the main characteristics of the pre-processing and cluster analysis system.

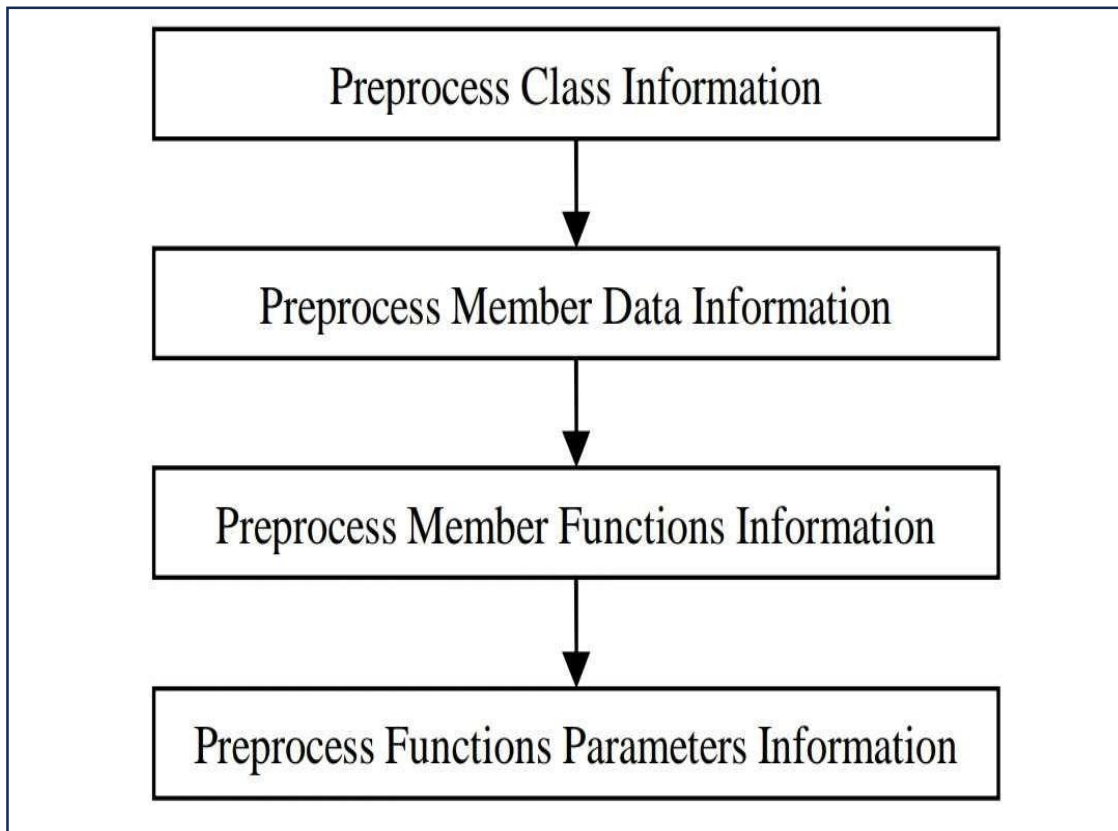
Each entity is described by atributes thus forming database tables



Pre-Processing Methodology

The pre-processing method extracts data from source code and stores these into appropriate tables. There were two major requirements for this:

i) Output should be stored in way facilitating clustering ii) Data processing should be fast. We use a top-down approach by processing top-level program data first, such as class information, and then lower-level data such as member functions, their parameters, and member data.



Pre-processing methodology

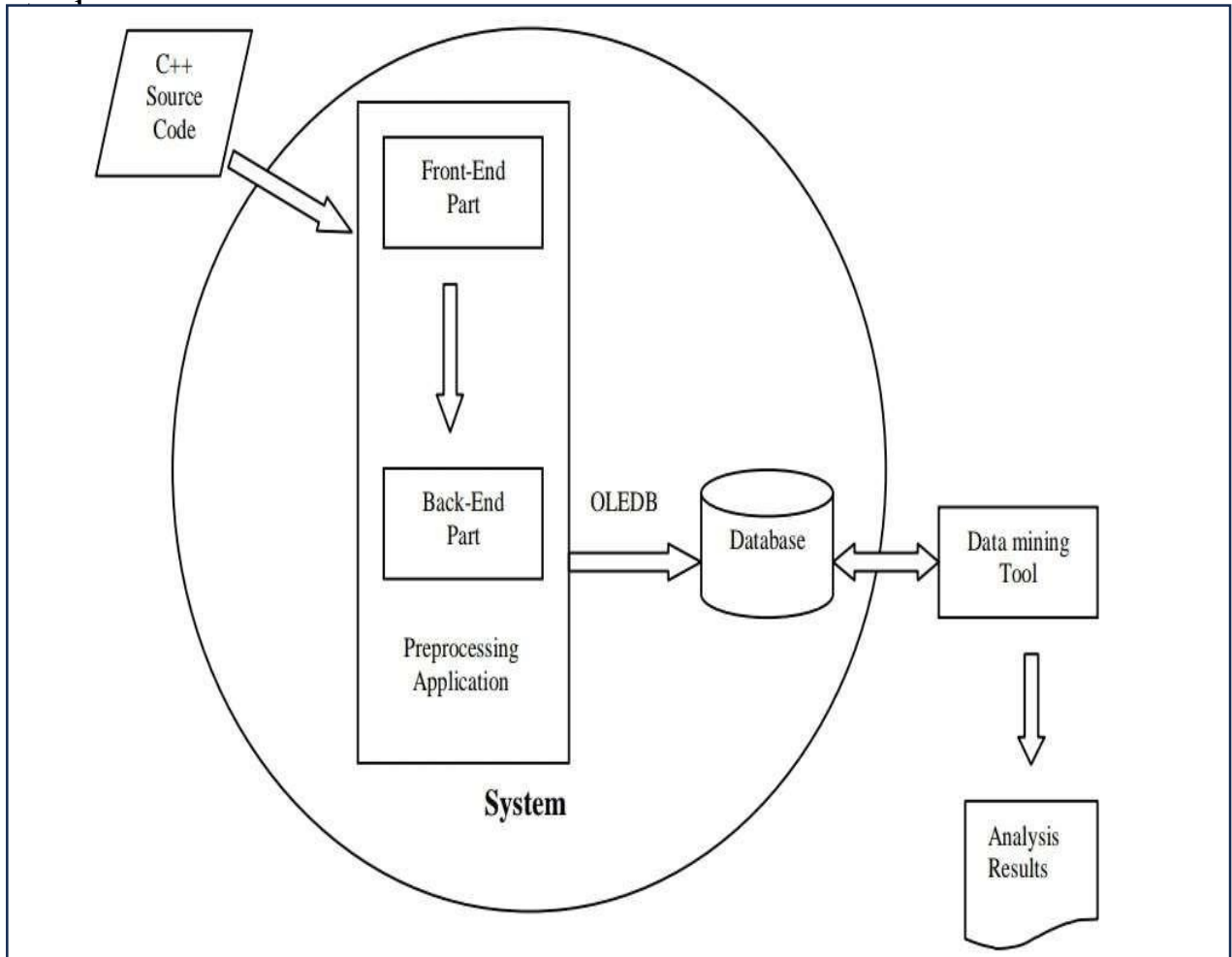
More specifically, we first extract information that describes the class entity, such as class handle, its super class name if it exists, and the number of the member data and functions. Then information that describes the member data of a class is extracted, including variable name, type and category, such as public, protected, private, as well as information describing whether the

variable is static, a pointer or user-defined. After that, we extract information related to class member functions, such as name, return type and category (public, protected, private), as well as number of parameters if any. Finally information related to the parameters of class member functions is extracted including name, type and use (by value, by reference). An outline of this methodology is illustrated

Aspects of the Proposed Framework

Sections §3.1-2 presented two fundamental concepts about the proposed framework: the input model and the pre-processing methodology. This section describes aspects of the framework regarding outcome utilization. More specifically all the information required by the methodology, as defined by the input data model, can be found at the header files of standard C++ systems. We scan these files and populate relevant database tables. We then use IBM's Intelligent Miner™ demographic clustering on these tables to identify patterns concerning the system structure and its components' general characteristics. These characteristics can be qualitative like the name of the superclass that a class inherits from, the category (public protected, private) of a member function and so on. They can also be quantitative, such as the number of member function parameters. We have experimented using various clustering schemes in order to identify correlated entities such as classes, functions and member data, based on similarities on their attributes as defined by the input model. Results are briefly presented and discussed in section 4.

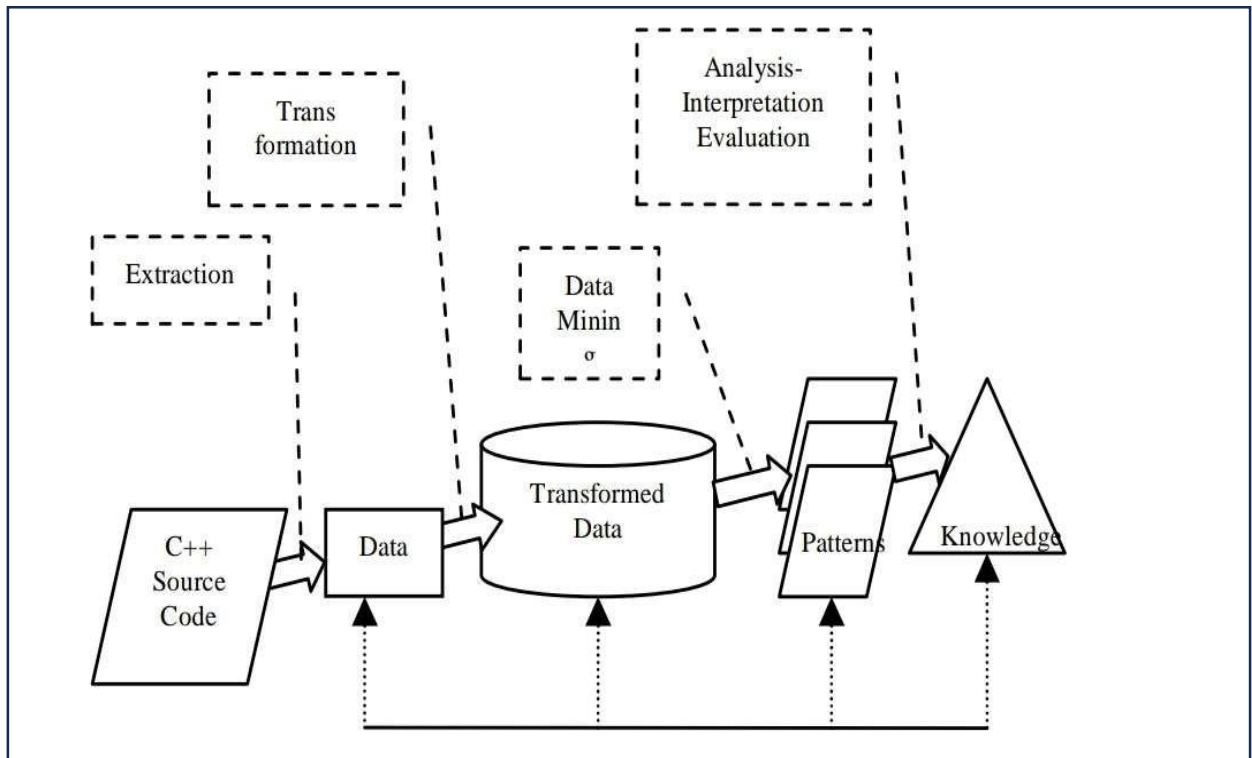
Collection is made of source code rather than more "conventional"



: An overview of the system

An overview of the system

The first step of this process (Extraction) involves parsing the code to extract data modeling program elements and their attributes. The second step (Transformation) transforms the extracted data in order to store these in relational database tables suitable for clustering. The third step (Data Mining) applies clustering in search for patterns of interest. Patterns are then interpreted and analyzed. The process is an iterative one and interim results or findings can be feedback to a previous stage.



Knowledge discovery in source code

Result Evaluation

The proposed framework was evaluated in terms of accuracy and ability to capture knowledge relevant to software maintenance activities, using three open source applications.

Two of the applications, CAccessReports and CompDB, are created with the help of Microsoft Foundation Classes (MFC) and can be downloaded from [19]. The other application, FlightGear Flight Simulator, is an open source flight simulator that can be downloaded from [19].

The actual structure of these applications is Front-End Part Database C++ Source Code Data mining Tool Analysis Results Back-End Part

Preprocessing Application System OLEDB Transformed Data Patterns
Knowledge Extraction Transformation Data Mining
Analysis Interpretation Evaluation C++ Source Code Data compared to
the outcome of the analysis of their respective input models.

The output should be valid, novel and useful to the system maintainer.
The following subsections discuss separately the outcomes of our
empirical experimentation with these applications.

The First Case Study

CAccessReport is a small-medium size application with 53 public classes, and 2812 functions that have 1614 parameters in total. 4.1.1 Class Analysis. The classes of this application have many similarities as almost all of them (52 out of 53) inherit from one class: COleDispatchDriver and have only public member functions. Therefore, only attributes describing the number of public functions and the class handle were of importance in forming clusters. As a result clusters are characterised only by the number of their member functions

Member Functions Analysis. There are two significant characteristics of the member functions of this program: the first is that all of them are public and the second is that almost half of them contain parameters. They were grouped in three clusters. The first cluster, representing 45.82% of the population, consists of public functions with parameters. These functions either have no return type or they return void. Therefore, it can be concluded that this cluster includes the constructors of the system's classes and functions that usually set values to these. The second cluster, representing 34.12% of the population, consists of public functions that have no parameters. Half of these functions return the type CString, which encapsulates a character string. The third cluster, representing 20.06% of the population, consists of public functions, 11.17% of which have no parameters at all, while the remaining 88.83%

have. Almost half of these functions return the following types VARIANT, which is a self-described data type that facilitates data passing, and LPDISPATCH, which accesses the underlying pointer of the COleDispatchDriver object

Member Functions Parameters Analysis. Most of the member function parameters are passed by value. They were grouped in three clusters: The first cluster (42.44% of the population) consists of parameters that are passed by value and originate from the following types: LPCSTR, which is a constant pointer to a string, LPDISPATCH and pointers of type VARIANT. The second cluster (41.57%) also consists of parameters that are passed by value, most of which originate from the types bool, short and long. The third cluster (15.99%)

Class Analysis.

Entities extracted from this program formed three clusters. The first cluster represents 38.89% of the population, and consists of classes that all inherit from another class. Their respective superclasses are: i) CStack, which encapsulates the stack control. ii) CView, which a view class is derived from. iii) CMDIFrameWnd, which provides a main frame window for Multiple Document Interface (MDI) applications. iv) CMDIChildWnd, which provides child windows for an MDI application.

Classes in this cluster are related logically, as they represent components of the document/view architecture implemented by this program. The second cluster, represents 33.33% of the population, and consists of classes, amongst which, two do not inherit and four do. The

respective superclasses of those who inherit are: i) CStringArray, which is an array of the String type. ii) CListBox, which encapsulates the list box control. iii) CDocument, which is the class where the document of an MFC application (like CompDB) derives from. iv) CListCtrl, which displays a graphical list items. The classes in this cluster do not have a strong logical correlation. There is only one class representing a component included in the document/view architecture, two others represent control classes, and another represents a shape of the MFC collection.

Member Data Analysis.

The member data of this program's classes are either public or protected. Three clusters were formed. The first cluster represents 59.38% of the population and consists of protected members, none of which is a pointer. Almost half of the member data of this class (Fig. 4.1) belong to two classes. The types of the member data vary. The more predominant are: i) int ii) CString, which encapsulates a character string. iii) CFont, which wraps the Windows font object and API functions for creating and managing fonts. iv) CGridCtrl, which is a control



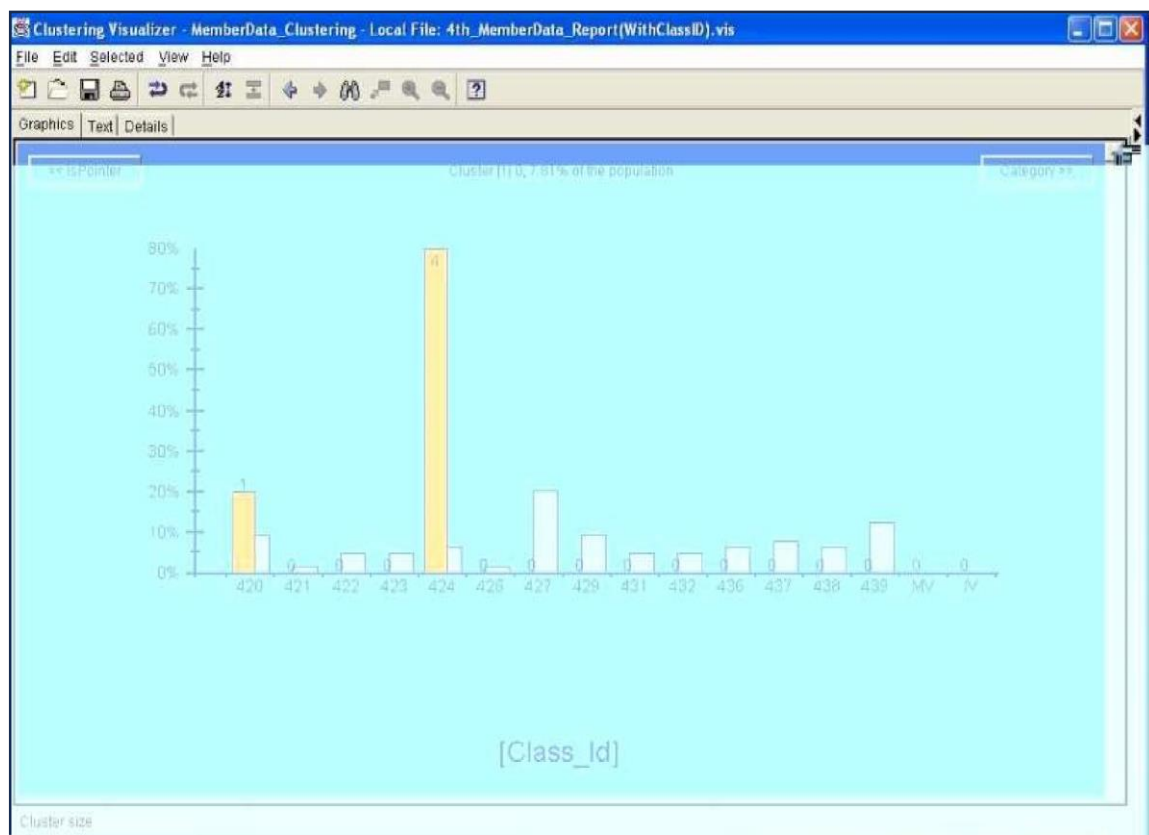
: Member data classes of CompDB, 1 st cluster

The second cluster represents 32.81% of the population and consists of public members none of which is a pointer. Member data of this cluster mostly belong to three classes. There is a clear logical connection among member data of this cluster as the majority of it belongs to classes that are derived from the CDialog class. Types of the member data vary. The more predominant are: i) enum ii) CString, which encapsulates a character string. iii) CButton, which wraps a standard Windows pushbutton.

The third cluster represents 7.81% of the population and consists of public and protected members which are all pointers. This is the most important logical relation between the member data of this cluster, which only belongs to two classes (Fig. 4.2). The types of the member data are

different. The more predominant are: i) CPen, which wraps the Windows pen object and includes API func ons for crea ng pens as member func ons. ii) CBrush, which wraps the Windows brush object and API func ons for crea ng brushes.

Member Func ons Analysis. Class member func ons were grouped in three clusters.



: Member data classes of CompDB, 3 rd cluster

The first cluster represents 34.38% of the population and consists of public and protected functions. The return types of these vary, the most predominant being `afx_msg void` and `afx_msg int`. Most of the member functions of this cluster belong to four classes. The second cluster represents 33.20% of the population.

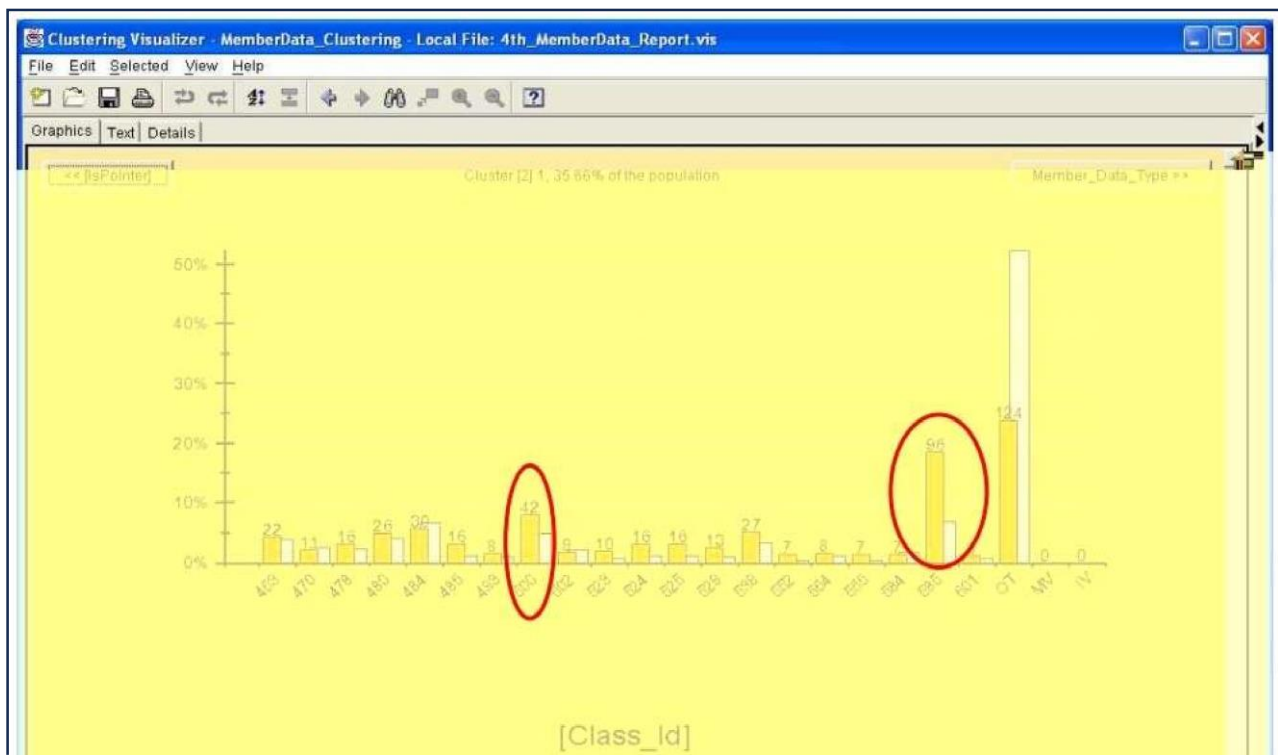
The return types of these vary, the most predominant being `void` and `bool`. Most of the member functions of this cluster belong to four classes, two of which are the same as in the first cluster. The third cluster represents 32.42% of the population. Almost half of the functions of this cluster do not have a return type. This indicates that they are either the constructors or the destructors of the classes they belong to. Among member functions that have a return type, the most predominant one is `bool`. Most of the member functions of this cluster belong to four classes, three of which are the same as in the second cluster and only one similar to these in the first cluster

Member Functions Parameters Analysis. Member function parameters of classes were grouped into three clusters:

The first cluster represents 42.98% of the population and consists of parameters passed by value. The return types of these vary, the most predominant being pointers of type `char`, `int` and pointers of type `CDC`,

which is a class that encapsulates device-context support. The second cluster represents 42.55% of the population and consists of parameters passed by value. The return types of these vary, the most predominant being char, UINT, which is an unsigned 32-bit integer and COLOREF, which is a 32-bit integer that holds a colour. The third cluster represents 14.47% of the population and consists of parameters passed by reference. The return types of these vary, the most predominant being CDUMPCONTEXT, which is a class that its objects provide several diagnostic messages and _CONNECTIONPTR, which is a class that its objects are pointers to a Connection Interface

Member Data Analysis. Clustering member data encompasses three clusters. The first cluster represents 38.34% of the population and consists of private member data one third of which are pointers. Most of them belong to class FGControls, which defines a standard interface to all flight simulation controls. The types of the member data vary. The more predominant are Bool, Int, SGPPPropertyNode and Float. The second cluster represents 35.66% of the population and consists of private data members, almost none of which are pointers. Most of them belong to classes FGInterface, which defines shared flight mode Member data classes in the 2nd cluster



question	optiona	optionb	optionc	optiond	answer	edit	delete
1 .choose the oldest programming language?	b language	c language	java language	java script language	a	edit	delete
2 .which is not a language of 8th schedules?	hindi	english	java language	all of these	c	edit	delete
3 .ponnian selvom is also known as...	samundra gupta	akbra	rajendra chola	raj raj chol	d	edit	delete
4 .which is known for its fragrance?	red sandalwood	white sandalwood	both	none	b	edit	delete
5 .ooty is located in which indian state?	bihar	punjab	tamilnadu	uttrakhand	c	edit	delete
6 .choose the immutable element in python.....	list	tuple	string	more than one	d	edit	delete
7 .how many seats of lower house are reserved for p.o.k ?	24	35	20	25	c	edit	delete

Navigate

[Home](#)
[About Us](#)
[FAQ](#)

FOLLOW US



Technical Support
777-234-098-127

777-234-098-127

Contact Form

Name

Email address

Message

Enter Title for blog

write blog

upload image

No file chosen

category

▾

question statement

choose the oldest programming language?

optiona

optionb

optionc

optiond

answer

 ▾

line €

is que

t a qu

direct

lit or 1

ge als

blog

writing page

question editing page

when admin want to edit question then he can click edit button before the question and question editing page will open where question and their options can be edited with update button which redirect to question database table.

If admin click on delete button, then after deleting question it will be redirected to question table. He can also upload a blog by upload button.

Models.py file

```
from django.db import models

class question(models.Model):
    qno=models.IntegerField(primary_key=True,auto_created=True)
    que=models.CharField(max_length=200,blank=True)
    optiona=models.CharField(max_length=100,null=True,blank=True)
    optionb=models.CharField(max_length=100,null=True,blank=True)
    optionc=models.CharField(max_length=100,null=True,blank=True)
    optiond=models.CharField(max_length=100,null=True,blank=True)
    ans=models.CharField(max_length=1)
```

this is model which is created to manage account creation

```
class user(models.Model):
    user_name=models.CharField(max_length=25,primary_key=True)
    email = models.EmailField(unique=True,null=False)
    password=models.CharField(max_length=25,null=False)
    gender=models.BooleanField(default=True)
```

this model is created for managing any particular message send by any user or student.

```
class message(models.Model):
    name=models.CharField(max_length=20)
    email=models.EmailField(primary_key=True,unique=True,null=False)

message=models.TextField()
def __str__(self):
    return self.name
```

View.py file of online exam

```
from django.shortcuts import render from django.http import
HttpResponse,HttpResponseRedirect from online_exam.models
import question,user,message import random
```

```
def set_question(request): return
render(request,'online_exam/set_question.html')
```

this views function save question to database when admin send new question through question setting page

```
def save_question(request):
    demo=question()
    demo.que=request.POST['question']
    demo.optiona=request.POST['optiona']
    demo.optionb=request.POST['optionb']
    demo.optionc=request.POST['optionc']
    demo.optiond=request.POST['optiond']
    demo.ans=request.POST['answer'] demo.save()
    return HttpResponseRedirect('http://localhost:8000/online_exam/view_question/')
```

this manage when admin wants to see question database

```
def view_question(request): try: if
request.session['sessionuser'] == 'admin':
    qlist=question.objects.all()
    return render(request,'online_exam/view_question.html',{'questions':qlist})
else:
    return HttpResponseRedirect('/online_exam/sign_in/')
except:
    return HttpResponseRedirect('/online_exam/sign_in/')
```


this views.py function run when admin edit
 question and click update button to save edit
 change

```
def edit_save(request):  
    n=int(request.POST['qnumber'])  
    Q=question.objects.get(qno=n)  
    Q.qno=n  
    Q.que=request.POST['question']  
    Q.optiona=request.POST['optiona']  
    Q.optionb=request.POST['optionb']  
    Q.optionc=request.POST['optionc']  
    Q.optiond=request.POST['optiond']  
    Q.ans=request.POST['answer']  
    Q.save()  
    return HttpResponseRedirect('http://localhost:8000/online_exam/view_question/')
```

 this views.py function run when admin click edit
 button before question

```
def edit_question(request):  
    try:  
        if  
        request.session['sessionuser'] == 'admin':  
            n=int(request.GET['qno'])  
            Q=question.objects.get(qno=n)  
            return  
            render(request,'online_exam/edit_question.html',{'question':Q})  
    else:  
        return  
        HttpResponseRedirect('/blog/main_blog/')  
    except:  
        return HttpResponseRedirect('/online_exam/sign_up/')
```

**this views.py function run when admin click delete
button before question**

```
def delete_question(request):
    try:
        n=int(request.GET['qno'])      ques=question.objects.get(qno=n)      ques.delete()
    return HttpResponseRedirect('http://localhost:8000/online_exam/view_question/')    except:
    return HttpResponseRedirect('http://localhost:8000/online_exam/sign_in/?error=2')
```

**this views.py function is for providing interface
after sign in**

```
def home(request):    try:        if
request.session['sessionuser']:
    return        render(request,'online_exam/home.html')
else:
    return    HttpResponseRedirect('/online_exam/sign_in/')
except KeyError:
    return    HttpResponseRedirect('/online_exam/sign_in/')
except:
    return HttpResponseRedirect('/online_exam/sign_in/?error=2')
```

```
def create_admin():    demo=user()
demo.user_name='admin'
demo.password='admin123'
demo.email='admin123@gmail.com'
demo.gender='1'    demo.save()
```

this views.py function run when admin logout

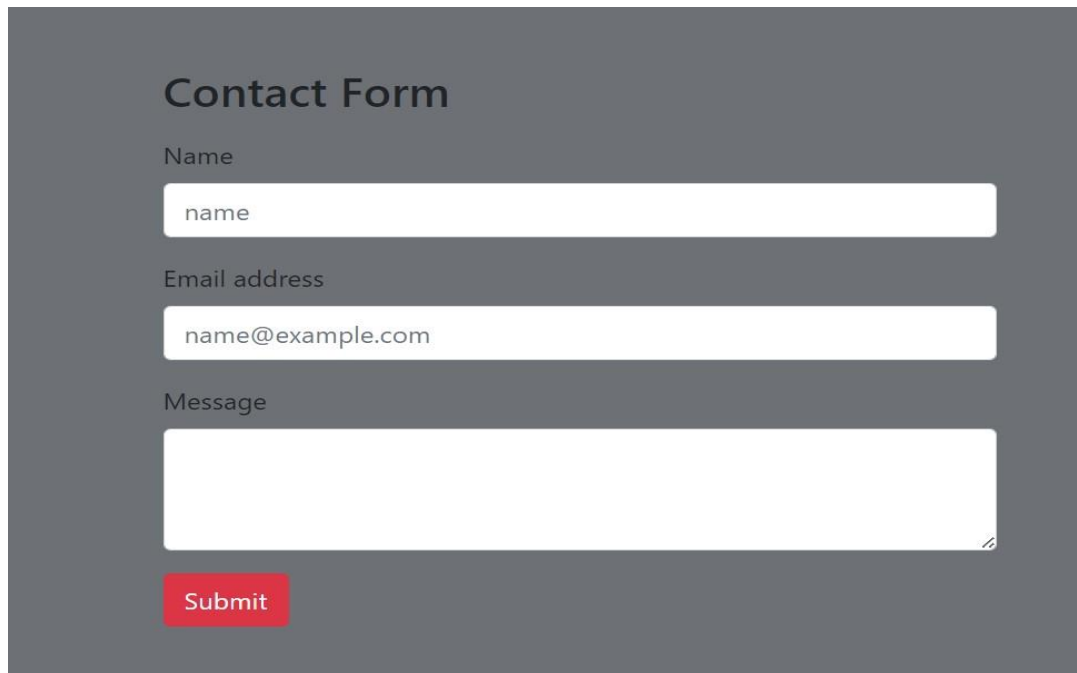
```
def log_out(request):
    request.session.clear()

    return HttpResponseRedirect('/online_exam/sign_in/')
```

this views.py function run when any user send message through footer section

```
def s_message(request):
    try:
        demo=message()
        demo.name=request.POST['name']
        demo.email=request.POST['email']
        demo.message=request.POST['message']
        demo.save()
        return
    except:
        return HttpResponseRedirect('some error occured .. try after some time')
```

when any user send message through contact form as shown in picture then the above views.py function will run .



The image shows a contact form titled "Contact Form" on a dark grey background. It contains three input fields: "Name" with the placeholder text "name", "Email address" with the placeholder text "name@example.com", and "Message" which is a larger text area. Below the fields is a red "Submit" button.

template of online exam module

this is the template page for admin and student which dynamically show content based on user.

```
{% extends "online_exam/base.html" %}
{% block title %}home{% endblock %}
```

```

{% block content %}
{% if request.session.sessionuser == 'admin' %}

<!-- navigator of admin-->
<ul class="nav justify-content-end">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href=
"http://localhost:8000/online_exam/set_question/">set new question</a>
  </li>
  <li class="nav-item">
    <a class="nav-link link-warning" href=
"http://localhost:8000/online_exam/view_question/">view question</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/blog/create_blog/">create_blog</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/online_exam/log_out/">log_out
</a>
  </li>

</ul>
<!-- Admin Dashboard -->
<div class="container mt-4">
  <!-- Admin Options Section -->
  <div class="admin-options">
    <h2>Welcome to <span id="welcome" style="color: blueviolet;"
></span></h2>
    <p>Welcome, {{request.session.sessionuser |capfirst}}!</p>

    <!-- Create Blog Option -->
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Create Blog</h5>
        <p class="card-text">Compose and publish new blog posts
for the community.</p>
        <a href="/blog/create_blog/" class="btn btn-primary">
Create Blog</a>
      </div>
    </div>

    <!-- Question Management Options -->
    <div class="card mt-3">
      <div class="card-body">

```

```

        <h5 class="card-title">Question Management</h5>
        <p class="card-text">Manage questions for online tests.
</p>
        <a href="/online_exam/set_question/" class="btn btn-
success">Create Question</a>
        <a href="/online_exam/view_question/" class="btn btn-
danger">View Question</a>
        <a href="#edit-question" class="btn btn-warning">Manage
discussion forum</a>
    </div>
</div>

<!-- Other Admin Options -->
<!-- Add more admin-specific options here -->

</div>
</div>

{% else %}

<!-- navigator of user-->
<ul class="nav justify-content-end">
    <li class="nav-item">
        <a class="nav-link active" aria-current="page" href=
"/online_exam/start_test/">start_test</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/online_exam/log_out/">log-out</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/blog/create_blog/">create_blog</a>
    </li>
</ul>

<!-- user dashboard -->
<div class="container mt-4">
    <!-- students Options Section -->
    <div class="admin-options">
        <h2>Welcome to <span id="welcome" style="color: blueviolet;"
></span></h2>
        <p>Welcome, {{request.session.sessionuser |capfirst}}!</p>

        <!-- Create Blog Option -->
<div class="card">
    <div class="card-body">
        <h5 class="card-title">Create Blog</h5>

```

```

        <p class="card-text">Compose and publish new blog posts
for the community.</p>
        <a href="/blog/create_blog/" class="btn btn-primary">
Create Blog</a>
    </div>
</div>

<!-- Question Management Options -->
<div class="card mt-3">
    <div class="card-body">
        <h5 class="card-title">Resource Management</h5>
        <p class="card-text">Manage Resources.</p>
<a href="/online_exam/start_test/" class="btn btn-
success">Start Test</a>
        <a href="#" class="btn btn-danger">post question on forum
</a>
        <a href="/notes/courses/" class="btn btn-warning">Explore
Notes </a>
    </div>
</div>

<!-- Other Admin Options -->
<!-- Add more admin-specific options here -->
{% endif %}

</div>

</div>
<!--cdn for auto text typing-->
<script src="https://unpkg.com/typed.js@2.1.0/dist/typed.umd.js"
></script>
<script>
    var typed = new Typed('#welcome', {
strings: ['RESOURCE MANAGEMENT', ],
    typeSpeed: 50,
backspeed:80,
loop:true    });
</script>

{% endblock %}

```

This template page is passed to views.py (def set_question(request) function) and this template page has form which send question data to save_question(request): function of views.py which

```

ultimately      save      data      in      database.
"Set_question.html"
{% extends "online_exam/base.html" %}
{% block title %} set_question{% endblock %}
{% block content %}
{% load static %}
<link rel="stylesheet" href="{% static 'css/question.css' %}">
{% if request.session.sessionuser == 'admin' %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end list-group list-group-horizontal-sm">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page" href=
"http://localhost:8000/online_exam/set_question/">set new question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link link-warning" href=
"http://localhost:8000/online_exam/view_question/">view question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href=
"http://localhost:8000/online_exam/log_out/">log_out</a>
        </li>
    </ul>

{% else %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end list-group list-group-horizontal-sm">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page" href=
"http://localhost:8000/online_exam/start_test/">start_test</a>
        </li>
        <li class="nav-item">
            <a
                class="nav-link"
                href=
"http://localhost:8000/online_exam/log_out/">log-out</a>
        </li>
    </ul>

```

```

        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
    </ul>
{% endif %}

<!--edit question-->
<div class="container-fluid">
    <div class="row">
        <div class="col-12 col-sm-12">

            <form action="http://localhost:8000/online_exam/save_question/"
method="post">
                {% csrf_token %}
                <div class="title">
question statement</div>
                <div class="data">
                    <textarea name="question" id="" cols="50" rows="5"
placeholder="enter question"></textarea>
                    </div>
                <div class="title">optiona</div>
                <div class="data">
                    <input type="text" name="optiona">
                </div>
                <div class="title">optionb</div>
                <div class="data">
                    <input type="text" name="optionb">
                </div>
                <div class="title">optionc</div>
                <div class="data">
                    <input type="text" name="optionc">
                </div>
                <div class="title">optiond</div>
                <div class="data">
                    <input type="text" name="optiond">
                </div>
                <div class="title">answer</div>
                <div class="data">
                    <select name="answer" id="">
                        <option value="a">a</option>
                        <option value="b">b</option>
                        <option value="c">c</option>
                        <option value="d">d</option>
                    </select>
                </div>
            </form>
        </div>
    </div>
</div>

```



```

        </select>
    </div>
    <input type="submit" value="save" class="btn btn-primary mt-2">

</form>
</div>
</div>
</div>

```

```
{% endblock %}
```

This template page is passed when admin request to view question database

```

{% extends "online_exam/base.html" %}
{% block title %}question_database{% endblock %}
{% block content %}

{% if request.session.sessionuser == 'admin' %}

    navigator of admin nav bar code is removed for space management
-->
    {% else %}

    <!-- navigator of student same as which shown in home page -->

    {% endif %}

    <!--fetching question from question_database-->
    <div class="table-responsive">

        <table class="table table-striped table-hover">
            <tr>
                <th>question</th>
                <th>optiona</th>
                <th>optionb</th>
                <th>optionc</th>
                <th>optiond</th>
                <th>answer</th>
                <th>edit</th>
                <th>delete</th>
            </tr>
            {% for q in questions %}
            <tr>
                <td class="que">{{forloop.counter}} .{{q.que}}</td>

```

```

        <td>{{q.optiona}}</td>
        <td>{{q.optionb}}</td>
        <td>{{q.optionc}}</td>
        <td>{{q.optiond}}</td>
        <td>{{q.ans}}</td>
        <td><a href=
"http://localhost:8000/online_exam/edit_question/?qno={{q.qno}}">edit</a></td>
        <td><a href=
"http://localhost:8000/online_exam/delete_question/?qno={{q.qno}}">delete</a></td>

    </tr>

    {% endfor %}

</table>
</div>
{% endblock %}

```

Template page which is used to render create_blog functionality called by def create_blog(request): function of blog views.py file. This template has form which send blog data for storage to def save_post(request): function of same.

```

{% extends "online_exam/base.html" %}
{% block title %}create_blog{% endblock %}
{% block content %}
{% if request.session.sessionuser == 'admin' %}

<!-- navigator of admin-->
<ul class="nav justify-content-end">
    <li class="nav-item">
        <a class="nav-link active" aria-current="page" href=
"http://localhost:8000/online_exam/set_question/">set new question</a>
    </li>
    <li class="nav-item">
        <a class="nav-link link-warning" href=
"http://localhost:8000/online_exam/view_question/">view question</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/blog/create_blog/">create_blog</a>
    </li>
    <li class="nav-item">
        <a class="nav-link"
href="/online_exam/log_out/">log_out</a>

```

```

        </li>

    </ul>

{% else %}

<!-- navigator of admin-->
<ul class="nav justify-content-end">
    <li class="nav-item">
        <a class="nav-link active" aria-current="page" href=
"/online_exam/start_test/">start_test</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/online_exam/log_out/">log-out</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/blog/create_blog/">create_blog</a>
    </li>
</ul>
{% endif %}

<!--blog writing-->
<div class="container">
    <form action="/blog/save_blog/" method="post" enctype=
"multipart/form-data">
        {% csrf_token %}
        <div class="title pl-4 ">Enter Title for blog</div>
        <div class="data pl-4">
            <textarea name="title" id="" cols="40" rows="3"></textarea>
        </div>
        <div class="title pl-4">write blog</div>
        <div class="title pl-4"><textarea name="content" id="" cols=
"40" rows="25"></textarea></div>
        <div class="title mt-4 pl-4">upload image</div>
        <div class="data pl-4">
            <input type="file" name="picture" >
        </div>
        <div class="title mt-3 pl-4">category</div>
        <div class="data pl-4">
            <select name="category" >
                {% for c in cat %}

                    <option value="{{c.no}}">{{c.name}}</option>
                {% endfor %}
            </select>

```

```

        </div>
        <input type="submit" value="upload" class="btn btn-success mt-
3">
    </form>
</div>
{% endblock %}

```

This template page is use to render edit question functionality .when admin enter edit button before question then this html page is used by def edit_question(request): function of views.py file of online exam.after editig this send data to edit_save(request): function to upde database.

```

        "Edit_question.html"
{% extends "online_exam/base.html" %}
{% block title %}edit_question{% endblock %}
{% block content %}
{% if request.session.sessionuser == 'admin' %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end list-group list-grouphorizontal-
sm">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page" href=
"/online_exam/set_question/">set new question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link link-warning" href=
"/online_exam/view_question/">view question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/online_exam/log_out/">log_out
</a>
        </li>

    </ul>

{% else %}

    <!-- navigator of admin-->

```

```

    <ul class="nav justify-content-end list-group list-group-horizontal-
sm">
    <li class="nav-item">
        <a class="nav-link active" aria-current="page" href=
"/online_exam/start_test/">start_test</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/online_exam/log_out/">log-out</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/blog/create_blog/">create_blog</a>
    </li>
</ul>
{% endif %}
<!--edit question section-->
<div class="container-fluid" <form
action="/online_exam/edit_save/" method="post">
    {% csrf_token %}
    <input type="hidden" name="qnumber" value="{{question.qno}}">
    <div class="title"> question statement</div>
    <div class="data">
        <textarea name="question" id="" cols="50" rows="5" >{{
question.que}}</textarea>
    </div>
    <div class="title">optiona</div>
    <div class="data">
        <input type="text" name="optiona" value=
"{{question.optiona}}">
    </div>
    <div class="title">optionb</div>
    <div class="data">
        <input type="text" name="optionb" value=
"{{question.optionb}}" >
    </div>
    <div class="title">optionc</div>
    <div class="data">
        <input type="text" name="optionc" value=
"{{question.optionc}}">
    </div>
    <div class="title">optiond</div>
    <div class="data">
        <input type="text" name="optiond" value=
"{{question.optiond}}">
    </div>
    <div class="title">answer</div>

```

```

    <div class="data">
        <select name="answer" id="">
            <option value="a" {% if question.ans == 'a' %}selected{%
endif %}>a</option>
            <option value="b" {% if question.ans == 'b' %}selected{%
endif %}>b</option>
            <option value="c" {% if question.ans == 'c' %}selected{%
endif %}>c</option>
            <option value="d" {% if question.ans == 'd'
%}selected{% endif %}>d</option>
        </select>
    </div>
    <div>
        <input type="submit" value="update" class="btn btn-
secondary mt-3">
    </div>

</form>
</div>
{% endblock %}

```

5.2 exam taking

student can give exams with multiple-choice question by login through sign in page. He can enjoy various features such as add blog, post question, start test.

When user login as student then resource management provide a different view and features while when an admin enters by login then it shows different view.

Welcome to RESOURCE M|

Welcome, Abhishek kumar!

Create Blog

Compose and publish new blog posts for the community.

Create Blog

Resource Management

Manage Resources.

Start Test

post question on forum

Explore Notes

Navigate

[Home](#)

[About Us](#)

[FAQ](#)

FOLLOW US



Technical Support

777-234-098-127

Contact Form

Name

name

Email address

name@example.com

Message

Submit

Views.py file for online_exam

when user sign in as student then above view is render dynamically by using this home function and home.html template file which is used for admin interface.

```
def home(request):
    try:
        if request.session['sessionuser']:
            return
        render(request, 'online_exam/home.html')
        else:
            return
        HttpResponseRedirect('/online_exam/sign_in/')
        except KeyError:
            return HttpResponseRedirect('/online_exam/sign_in/')
        except:
```

```

return HttpResponseRedirect('/online_exam/sign_in/?error=2') this
function execute when an student click on start test button
def start_test(request):    try:        if request.session['sessionuser']:
qlist=list(question.objects.all())                random.shuffle(qlist)
qpool=qlist[:5]                                return
render(request,'online_exam/start_test.html',{qpool:qpool})    else:
return HttpResponseRedirect('/online_exam/sign_in/')        except:
return HttpResponseRedirect('/online_exam/sign_in/')

```

(start_test.html)template file is used by
def start_test(request): function for
rendering

```

{% extends "online_exam/base.html" %}
{% block title %}test{% endblock %}
{% block content %}
{% if request.session.sessionuser == 'admin' %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page" href=
"http://localhost:8000/online_exam/set_question/">set new question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link link-warning" href=
"http://localhost:8000/online_exam/view_question/">view question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href=
"http://localhost:8000/online_exam/log_out/">log_out</a>
        </li>

```



```

    </ul>

    {% else %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page" href=
"http://localhost:8000/online_exam/start_test/">start_test</a>
        </li>
        <li class="nav-item">
            <a
                class="nav-link"
                href=
"http://localhost:8000/online_exam/log_out/">log-out</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
    </ul>
    {% endif %}

<div class="container-fluid">
    <div class="row pl-4">
        <div class="col-10">

            <form action="http://localhost:8000/online_exam/test_result/"
method="post">
                {% csrf_token %}
                {% for q in qpool %}
                <input type="hidden" name="qno{{q.qno}}" value="{{q.qno}}">
                <div class="question">{{forloop.counter}} .{{q.que}}</div>
                <div class="option"><input type="radio" name="ans{{q.qno}}" value="a"
                >{{q.optiona}}</div>
                <div class="option"><input type="radio" name="ans{{q.qno}}"
                value="b" >{{q.optionb}}</div>
                <div class="option"><input type="radio" name="ans{{q.qno}}"
                value="c" >{{q.optionc}}</div>
                <div class="option"><input type="radio" name="ans{{q.qno}}"
                value="d" >{{q.optiond}}</div>
                {% endfor %}
                <input type="submit" value="submit" class=" btn btn-primary mt-
                3" >
            </div>

```

```
        </div>
    </form>
</div>
{% endblock %}
```

the interface when a student clicks on start_test then randomly a chunk of 5 question comes from database in multiple choice questions. When one refresh page then questions will be changed.

- 1 .choose the oldest programming language?
 - b language
 - c language
 - java language
 - java script language
- 2 .choose the immutable element in python....
 - list
 - tuple
 - string
 - more than one
- 3 .how many seats of lower house are reserved for p.o.k ?
 - 24
 - 35
 - 20
 - 25
- 4 .which is not a language of 8th schedules?
 - hindi
 - engliish
 - java language
 - all of these
- 5 .ooty is located in which indian state?
 - bihar
 - punjab
 - tamilnadu
 - outtrakhand

submit

Navigate

- [Home](#)
- [About Us](#)
- [FAQ](#)

FOLLOW US

Technical Support

777-234-098-127

Contact Form

Name

Email address

Message

5.3 Result Analysis:

When a student submits the test then his test data is sent to test result function for result analysis, where it shows where we make a mistake and right wrong questions.

Views.py file of online_exam

```
def test_result(request):
    try:
        if
    request.session['sessionuser']:
    total_wrong=0
        total_write=0
```

```

attempted_ques=0                wq=[]
qlist=[]                        for k in request.POST:
if                               k.startswith('qno'):
qlist.append(int(request.POST[k]))
for n in qlist:                  try:
q=question.objects.get(qno=n)
if q.ans == request.POST['ans'+str(n)]:
total_write+=1                  else:

total_wrong+=1
wq.append(q)
attempted_ques+=1
except:                           pass
d={
                                'total_wrong':total_wrong ,
                                'total_wright':total_write ,
                                'attempted_ques':attempted_ques,
                                'wq':wq
}
                                return render(request,'online_exam/test_result.html',d)
except:
                                return HttpResponseRedirect(
'http://localhost:8000/online_exam/sign_in/')

```

template file which is used for result analysis

```

{% extends "online_exam/base.html" %}
{% block title %}test_result{% endblock %}
{% block content %}
{% if request.session.sessionuser == 'admin' %}

<!-- navigator of admin-->
<ul class="nav justify-content-end">
<li class="nav-item">
<a class="nav-link active" aria-current="page" href=
"http://localhost:8000/online_exam/set_question/">set new question</a>
</li>
<li class="nav-item">
<a class="nav-link link-warning" href=
"http://localhost:8000/online_exam/view_question/">view question</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/blog/create_blog/">create_blog</a>
</li>

```

```

        <li class="nav-item">
            <a class="nav-link" href=
"http://localhost:8000/online_exam/log_out/">log_out</a>
        </li>

    </ul>

    {% else %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page" href=
"http://localhost:8000/online_exam/start_test/">start_test</a>
        </li>
        <li class="nav-item">
            <a
                class="nav-link"
                href=
"http://localhost:8000/online_exam/log_out/">log-out</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
    </ul>
    {% endif %}

    <table class="table table-dark table-striped pl-3 mt-3">
        <tr>
            <th>total question:</th>
            <th>5</th>
        </tr>
        <tr>
            <th>attempted question</th>
            <th>{{attempted_ques}}</th>
        </tr>
        <tr>
            <th>wright
                question</th>
            <th>{{total_wright}}</th>
        </tr>
        <tr>
            <th>wrong
                question</th>
            <th>{{total_wrong}}</th>
        </tr>
    </table>
    <table class="table table-striped table-hover pl-3">

```

```
    <tr>
        <th class="text-center" style="color: red;"> Questions in
which you make mistake <h1>❗</h1></th>
    </tr>
```

```
{% for q in wq %}
<tr>
    <td>

    <div>{{forloop.counter}} .{{q.que}}</div>
    <div>{{q.optiona}} <input type="radio" {% if q.ans == 'a' %} checked
{% endif %}></div>
    <div>{{q.optionb}} <input type="radio" {% if q.ans == 'b' %} checked
{% endif %}></div>
    <div>{{q.optionc}} <input type="radio" {% if q.ans == 'c' %} checked
{% endif %}></div>
    <div>{{q.optiond}} <input type="radio" {% if q.ans == 'd' %} checked
{% endif %}></div>
    </td>
</tr>
{% endfor %}
<tr>
    <td class="text-center">
        <a
href="/online_exam/start_test/">test again</a>
    </td>
</tr>
</table>
{% endblock %}
```

total question:	5
attempted question	5
wright question	2
wrong question	3

Questions in which you make mistake



1 .choose the oldest programming language?

- b language
- c language
- java language
- java script language

2 .choose the immutable element in python.....

- list
- tuple
- string
- more than one

3 .which is not a language of 8th schedules?

- hindi
- english
- java language
- all of these

test again

Navigate

- [Home](#)
- [About Us](#)
- [F.A.Q](#)

FOLLOW US



Technical Support 777-234-098-127

Contact Form

Name

Email address

Message

Blog system

The Blog System is a key feature of the Resource Management Project, providing users with a collaborative platform for sharing insights, knowledge, and fostering community engagement. Developed within the Django framework, this module enhances the educational experience by facilitating communication, information exchange, and collaborative learning.

Thank You