

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI



A Mini Project Report

On

“Cyber-Cafe Management System”

Submitted as part of curriculum 2021 scheme with course code 21CSMP67

BACHELOR OF ENGINEERING

In

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Submitted By

R SHREE HARI	4GM21CS071
RAGASUDHA G	4GM21CS074
SUMITH M SHIRALLI	4GM21CS110
YASHASWINI M O	4GM21CS125

Project Guide

Mrs. Nayana K

Assistant Professor

Co - Ordinators

Mr. Kotreshi S N

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GM INSTITUTE OF TECHNOLOGY, DAVANGERE



(Affiliated to VTU, Belagavi, Approved by AICTE -New Delhi & Govt. of Karnataka)

(Accredited by NBA New Delhi, Valid up to 30.06.2025)

2023-2024

Srishyla Educational Trust (R), Bheemasamudra

GM INSTITUTE OF TECHNOLOGY, DAVANGERE

(Affiliated to VTU, Belagavi, Approved by AICTE -New Delhi & Govt. of Karnataka)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA New Delhi, Valid up to 30.06.2025)



CERTIFICATE

Certified that the Mini Project titled “**Cyber Cafe Management System**” is a bonafide work carried out by **R SHREE HARI (4GM21CS071)**, **RAGASUDHA G (4GM21CS074)**, **SUMITH M SHIRALLI (4GM21CS110)**, **YASHASWINI M O (4GM21CS125)** as per curriculum scheme 2021 with course code 21CSMP67 of Bachelor of Engineering in the Department of Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi, during the year 2023-24. The Mini Project report has been approved as it satisfies the academic requirements with respect to the Mini Project work prescribed for Bachelor of Engineering Degree.

Project Guide

Mrs. Nayana K

Coordinator

Mr. Kotreshi S N

Head of the Department

Dr. B N Veerappa

ACKNOWLEDGEMENT

The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible.

We would like to express our gratitude to our **Principal, Dr. Sanjay Pande M B** for providing us a congenial environment for engineering studies and also for having showed us the way to carry out the Mini Project work.

We consider it a privilege and honour to express our sincere thanks to **Dr. B N Veerappa**, Professor and Head, Department of Computer Science and Engineering for his support and invaluable guidance throughout the tenure of this Mini Project work.

We would like to thank our Guide **Mrs. Nayana K**, Assistant Professor, Department of Computer Science and Engineering for his support, guidance, motivation, encouragement for the successful completion of this Mini Project work.

We intend to thank all the teaching and non-teaching staffs of our Department of Computer Science and Engineering for their immense help and co-operation.

Finally, We would like to express our gratitude to our parents and friends who always stood by us.

Student Name

R SHREE HARI	4GM21CS071
RAGASUDHA G	4GM21CS074
SUMITH M SHIRALLI	4GM21CS110
YASHASWINI M O	4GM21CS125



Vision

To build excellent Technocrats in Computer Science and Engineering by continuously striving for excellence in IT industry to meet the challenges of society.

Mission

1. *To train students by adopting effective teaching-learning approach.*
2. *To establish collaborative learning approach with Industry and Professional bodies.*
3. *To develop engineers with Professional-Social ethics and creative Research Culture.*

Program Educational Objectives

1. Graduates able to apply the knowledge of Basics Science and Core Computer science to analyze and solve real world problems.
2. Graduates possess professional skills needed for IT employment and pursue higher education in Computer Science and Engineering.
3. Graduates engage in life-long learning and adapt to changing Environment.
4. Graduates who can succeed as an individual or team leader in multidisciplinary avenues.

Program Specific Outcomes

Graduates of Computer science & engineering are able to:

1. Understand basic principles of programming and core concepts of computer science.
2. Design and analyze software solutions for real world problems using computational models.
3. Develop Applications using C, Object Oriented Concepts, Computer Graphics, Database Management System, Web Programming, Machine Learning and Mobile Application Development to meet current industry and entrepreneurial requirements.



Programme outcomes of Computer Science & Engineering

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The purpose of Cyber Cafe Management System is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with. Cyber Cafe Management System, as described above, can lead to error free secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus, it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. Moreover, the system's intuitive interface simplifies navigation for both staff and customers, making it easy to manage accounts, initiate sessions, and process payments. Detailed reporting capabilities provide insights into usage patterns and financial performance, enabling informed decision-making and strategic planning. This Django-based solution not only improves operational efficiency but also elevates the overall customer experience by reducing wait times and ensuring accurate billing. As a result, cyber cafe owners can focus on providing a better service environment, ultimately driving customer satisfaction and business growth.

CONTENTS

	Page No.
Chapter 1: Introduction	01 - 02
1.1 Problem Statement	
1.2 Objectives	
1.3 Proposed Solution	
Chapter 2: Django and Full Stack Development	03 – 08
2.1 History and Development	
2.2 Architecture	
2.3 Core Components	
2.4 Features	
2.5 Extensions and Versions	
2.6 Applications	
Chapter 3: Design and Implementation	09 - 24
3.1 Requirement Specifications	
3.2 Entity - Relationship Diagram	
3.3 Schema Diagram	
3.4 Implementation of Objective	
Chapter 4: Results	25 - 32
Chapter 5: Conclusion	33
Bibliography	34

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.2	Entity Relationship Diagram	10
3.3	Schema Diagram	11
4.1	Admin Login	25
4.2	Dashboard	25
4.3	Admin Profile	26
4.4	Change Password	26
4.5	Add Computer	27
4.6	Manage Computer	27
4.7	Update Computer	28
4.8	Add Users	28
4.9	Manage New User Details	29
4.10	Update New Users Details	29
4.11	Manage Old Users Details	30
4.12	View Od Users Details	30
4.13	Search Data	31
4.14	Between Reports	31
4.15	Snapshot of Chart Prepared	32

Chapter 1

INTRODUCTION

The “Cyber Cafe Management System” has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

“Cyber Cafe Management System” is a software package, which can be used in cyber cafés for managing the clients’ computer efficiently. Now a day’s cyber terrorism, which is mainly undergone through internet cafés, need to be tackled properly. Thereby, it is indeed necessary to store the valid information of the user who comes for internet access. The system being used, the time at which the user logs in and logs out should be recorded systematically. In this modern era, a number of people access the internet frequently by means of cyber cafes. For such frequent users, a prepaid account shall be maintained and discounted rates may be charged from them. While walkthrough users, who are less frequent, are charged a fixed rate. By using the LAN connections in the cyber cafes, we can automate this process very easily. This system comprises of the following modules: The Server/Admin Module The server module, which is handled by the administrator can create new accounts for prepaid users and also store the details of walkthrough customers.

1.1 PROBLEM STATEMENT:

The Cyber Cafe Management System aims to address the challenges of managing a cyber cafe efficiently by automating key processes such as user account management, session tracking, and resource allocation. In the present scenario the café owner keeps a paper book to keep track of user details. Manual processing of data is always time consuming and may commit more errors. There is much difficulty in allocating cabins to the users. Further reference to the user details is time consuming. Accuracy of such data makes the system unreliable and inefficient. Obviously there is need of an efficient system. By implementing this system, cyber cafe owners can streamline operations, reduce errors, enhance customer satisfaction, and optimize resource utilization, ultimately leading to a more organized and profitable business.

1.2 OBJECTIVES:

- The main objective of the Project on Cyber Cafe Management System is to manage the details of the Computer, Time, Customer, Customer details, Keypad and Payment.
- It manages all the information about Computer, Time, Customer, Customer details, Keypad and Payment.
- The project is totally built at administrative end and thus only the administrator is guaranteed the access.
- The purpose of the project is to build an application program to reduce the manual work for managing the Computer, Time, Payment, Customer.
- It tracks details about the Customer and Computers.

1.3 PROPOSED SOLUTION:

To address the operational challenges of running a cyber cafe, we propose the development of a Cyber Cafe Management System utilizing Django (a high-level Python web framework), Python programming language, and MySQL database.

In this project an attempt is made to design a computer system for the Cyber Cafe that makes the management of recording user details, internet usage and billing much easier. The objective of this software is to maintain the details of users, cabins and login history. Through this system we provide facility of prepaid and post-paid accounts respectively for Account Users and Walkthrough Users.

The Software powered by Python assures clear and efficient services to the agency. This easy-to-operate system helps to access and modify user details, provides efficient billing facility. The software is designed to provide Reliable and error free information. The database is driven by MySQL thus providing portability.

Chapter 2

DJANGO AND FULL STACK DEVELOPMENT

2.1 HISTORY AND DEVELOPMENT:

2.1.1 Django Framework: Django is a high-level Python web framework that promotes rapid development and clean, structured design. It aims to simplify the creation of complex, database-driven websites by providing a comprehensive set of tools and functionalities out of the box. It encourages rapid development and clean, pragmatic design. It was created in the fall of 2003 by Adrian Holovaty and Simon Willison while working at the Lawrence Journal-World newspaper. The framework was publicly released under a BSD license in July 2005. The main goal behind Django was to simplify the process of building complex, database-driven websites. Its name was inspired by Django Reinhardt, a famous jazz guitarist.

2.1.2 Full Stack Development:

- **Early Days:** In the early stages of web development, the roles of front-end and back-end developers were more distinct. Front-end development involved HTML, CSS, and basic JavaScript, while back-end development involved server-side scripting languages like Perl, PHP, and early versions of databases.
- **Rise of JavaScript:** The introduction of JavaScript and the evolution of its frameworks (like jQuery, AngularJS, React, and Vue.js) significantly changed the landscape of front-end development.
- **Server-Side Technologies:** The backend saw the rise of powerful frameworks like Ruby on Rails, Django, and Node.js, enabling more dynamic and scalable server-side applications.
- **Full Stack Evolution:** With the growth of technologies and the need for more dynamic, interactive web applications, the role of the full stack developer emerged. These developers possess skills across the entire stack, allowing them to handle both the front-end and back-end aspects of web development.

2.2 ARCHITECTURE:

2.2.1 Django: Django follows the Model-Template-View (MTV) architectural pattern, which is a variation of the Model-View-Controller (MVC) pattern:

- **Model:** This layer defines the structure of the database. It is responsible for business logic and data retrieval.
- **Template:** This layer controls how the data is presented to the user. It uses Django's templating language to create dynamic HTML.
- **View:** This layer contains the logic that handles user input and interacts with the model and template to produce a response.

Additionally, Django uses a URL dispatcher to map URL patterns to views, ensuring a clean and intuitive URL scheme.

2.2.2 Full Stack Development:

- **Monolithic Architecture:** A single-tiered software application in which different components are combined into a single program from a single platform.
- **Microservices Architecture:** An architectural style that structures an application as a collection of loosely coupled services. It enhances modularity and allows the application to be more flexible and scalable.
- **Client-Server Architecture:** The client requests resources or services, and the server provides them. This model separates the user interface concerns from the data storage concerns.
- **MVC (Model-View-Controller):** A design pattern that separates an application into three main components: Model (data), View (UI), and Controller (logic).

2.3 CORE COMPONENTS:

2.3.1 Django:

- **Models:** Define the structure of the database and the relationships between data. Models are defined using Python classes and are mapped to database tables.
- **Views:** Handle the request/response logic. Views retrieve data from the model and pass it to the template.
- **Templates:** Manage the presentation layer using Django's template language, which allows embedding dynamic content within HTML.
- **Forms:** Provide an easy way to handle form data, including validation and rendering HTML form elements.
- **Admin Interface:** A built-in, customizable interface for managing site content. It is automatically generated based on the models defined in the application.
- **ORM (Object-Relational Mapping):** Allows developers to interact with the database using Python code instead of SQL queries. The ORM handles database queries, schema migrations, and relationships between tables.
- **Middleware:** Components that process requests and responses globally before they reach the view or before they are sent to the client. Middleware can be used for tasks such as authentication, caching, and session management.

2.3.2 Full Stack Development:

1. Front-End Technologies:

- **HTML/CSS:** Fundamental technologies for structuring and styling web pages.
- **JavaScript:** A versatile scripting language used for creating dynamic content.
- **Frameworks/Libraries:** React, Angular, Vue.js, etc., which help build modern, responsive UIs.

2. Back-End Technologies:

- **Server-Side Languages:** Node.js, Python, Ruby, PHP, Java, etc.
- **Frameworks:** Express.js (Node.js), Django (Python), Ruby on Rails (Ruby), Laravel (PHP), etc.
- **Databases:** SQL databases (MySQL, PostgreSQL) and NoSQL databases (MongoDB, Cassandra).

3. APIs: RESTful services and GraphQL for communication between front end and back end.
4. Version Control: Git for managing code versions and collaboration.
5. DevOps Tools: Tools for deployment, continuous integration/continuous deployment (CI/CD), and containerization (Docker, Kubernetes).

2.4 FEATURES:

2.4.1 Django:

- **DRY Principle:** Django emphasizes the "Don't Repeat Yourself" philosophy, which helps reduce redundancy and improve code maintainability.
- **URL Routing:** Clean and dynamic URL configurations that allow developers to define URL patterns and map them to views.
- **Authentication:** Built-in user authentication system that handles user registration, login, logout, password management, and more.
- **Scalability:** Django can handle large volumes of traffic and is designed to scale horizontally across multiple servers.
- **Security:** Django includes built-in protections against common security threats such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking.
- **Internationalization:** Built-in support for multiple languages and localization, allowing developers to create multilingual applications easily.

2.4.2 Full Stack Development:

- **Versatility:** Ability to work on both client and server sides.
- **Proficiency in Multiple Technologies:** Knowledge of various programming languages, frameworks, and tools.
- **Efficient Problem-Solving:** Capability to troubleshoot and resolve issues across the entire stack.
- **Adaptability:** Keeping up with the latest trends and technologies in web development.
- **End-to-End Project Management:** Managing the complete lifecycle of a project from concept to deployment and maintenance.

2.5 EXTENSIONS AND VERSIONS:

2.5.1 Django:

- Django Packages: Extensible via third-party packages available on the Django Packages website. These packages extend Django's functionality and can be easily integrated into existing projects.
- Django REST Framework: An extension for building RESTful APIs. It provides a powerful and flexible toolkit for creating Web APIs.
- Versions: Django has a regular release cycle with Long Term Support (LTS) versions that receive extended support and security updates. Each major release introduces new features and improvements while maintaining backward compatibility.

2.5.2 Full Stack Development:

- Front-End Extensions: Libraries like Redux (for state management), Axios (for HTTP requests), and Bootstrap (for responsive design).
- Back-End Extensions: Middleware for handling authentication, authorization, and other functionalities.
- APIs: Third-party APIs to extend functionality, such as payment gateways (Stripe, PayPal) and social media integrations.
- Versions: Keeping up with the latest versions of languages, frameworks, and tools to leverage new features and improvements.

2.6 APPLICATIONS:

2.6.1 Django:

- Content Management Systems (CMS): Frameworks like Wagtail and Mezzanine are built on Django and provide robust content management capabilities.
- Social Media Platforms: Django powers social media platforms like Instagram, enabling them to handle massive amounts of user-generated content and traffic.
- E-commerce Sites: Frameworks like Saleor and Oscar use Django to build scalable and feature-rich e-commerce platforms.

- Scientific Computing: Django is used in projects that require robust backend support for scientific computing and data analysis.
- Educational Platforms: Many Massive Open Online Courses (MOOCs) and Learning Management Systems (LMS) are built using Django due to its scalability and ease of use.

2.6.2 Full Stack Development:

- E-Commerce: Developing scalable online stores with features like product listings, shopping carts, payment processing, and order management.
- Social Media Platforms: Building platforms with real-time communication, user profiles, content sharing, and networking features.
- Content Management Systems (CMS): Creating systems for managing digital content, such as blogs, news websites, and corporate sites.
- Enterprise Applications: Developing internal tools and applications for businesses, including CRM systems, ERP solutions, and HR management software.
- Mobile App Development: Using frameworks like React Native or Flutter to build cross-platform mobile applications with a shared codebase.
- Data-Driven Applications: Creating applications that handle large datasets, perform data analysis, and provide insights through dashboards and visualizations.

Chapter 3

DESIGN AND IMPLEMENTATION

3.1 REQUIREMENT SPECIFICATION:

The software requirements for the Cyber Cafe platform using Django outline the essential tools and technologies needed to develop, deploy, and maintain a robust and user-friendly application. These requirements include a combination of programming languages, frameworks, databases, and development tools designed to ensure efficient functionality, secure transactions, and an optimal user experience.

Software Requirements for Full Stack Development Project of Online Grocery Shopping Using Django:

3.1.1 Software Requirements:

- Operating System: Windows 11
- Front End: Full Stack (HTML, CSS, JavaScript)
- Back End: Django
- Programming Language: Python
- Database: SQLite
- Browser Compatibility: Google Chrome, Microsoft Edge
- Integrated Development Environment (IDE): Visual Studio Code

Hardware Requirements for Full Stack Development Project of Online Grocery Shopping Using Django:

3.1.2 Hardware Requirements:

- Computer with a 1.1 GHz or faster processor
- Minimum 4GB of RAM or more
- 2.5 GB of available hard-disk space
- SSD with at least 100 GB of free space
- High-speed internet connection with a reliable network interface card
- Monitor with a resolution of at least 1080p

3.2 ENTITY – RELATIONSHIP DIAGRAM:

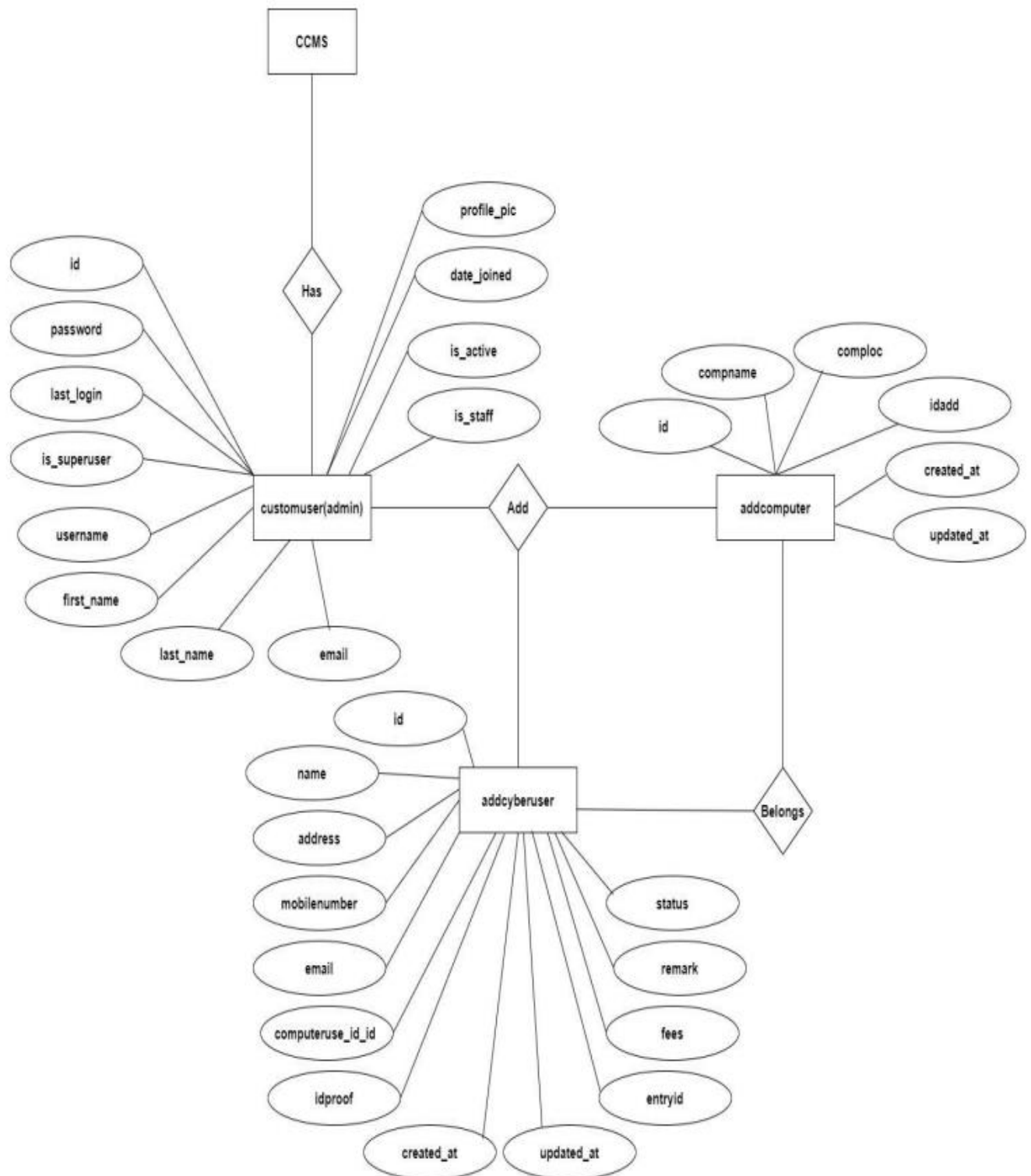


Fig 3.2 Entity – Relationship Diagram

3.3 SCHEMA DIAGRAM:

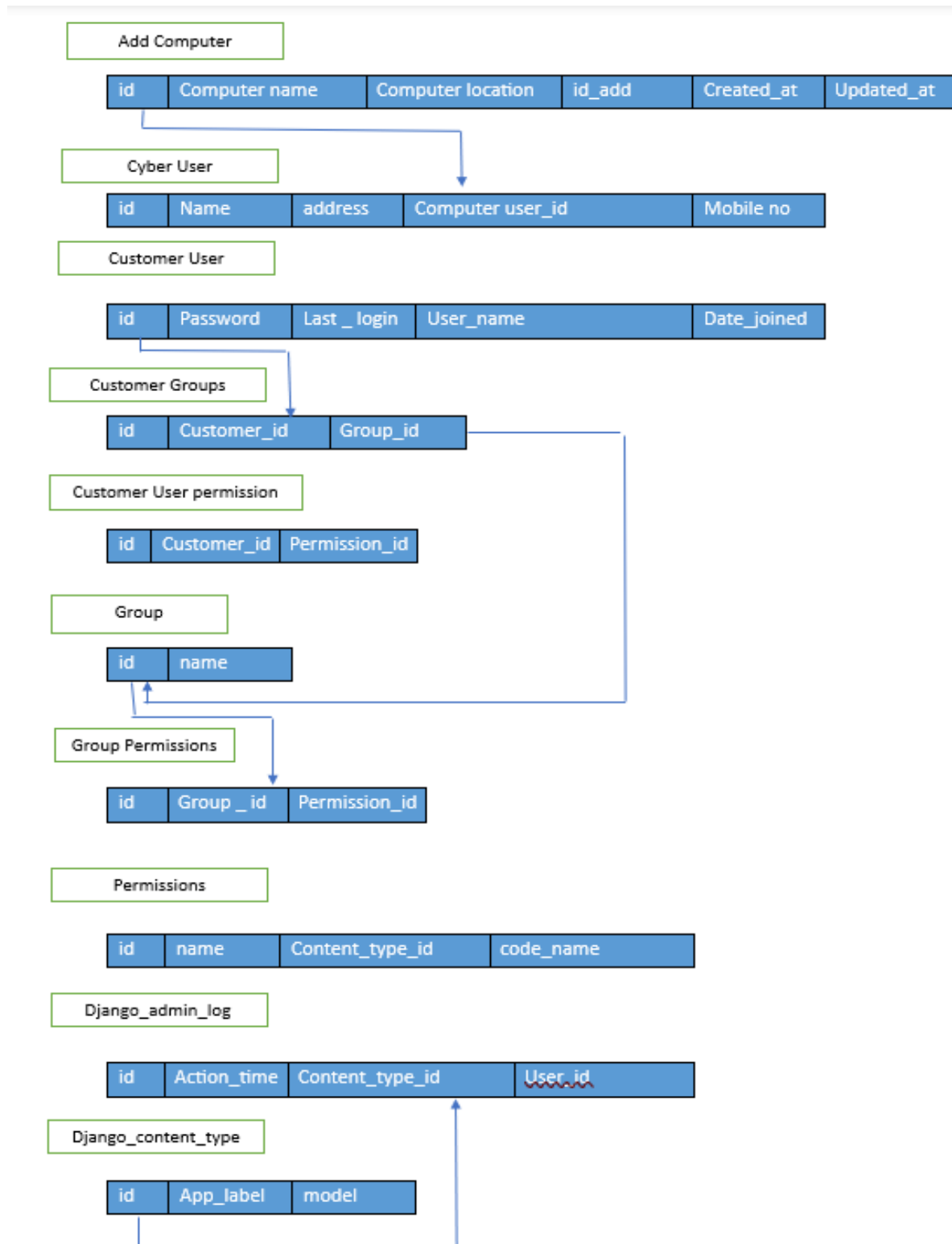


Fig 3.3' Schema Diagram

3.4 IMPLEMENTATION OF OBJECTIVE:

3.4.1 Views.py:

```
from django.shortcuts import render,redirect,HttpResponse

from authentication.EmailBackEnd import EmailBackEnd

from django.contrib.auth import logout,login

from django.contrib import messages

from django.contrib.auth.decorators import login_required

from authentication.models import CustomUser, AddComputer,AddCyberUser

from django.contrib.auth import get_user_model

import random


# Create your views here.

def BASE(request):

    return render(request,'base.html')


def LOGIN(request):

    return render(request,'login.html')


def doLogin(request):

    if request.method == 'POST':

        user = EmailBackEnd.authenticate(request,

                                         username=request.POST.get('email'),

                                         password=request.POST.get('password'))
```

```
        )

    if user!=None:

        login(request,user)

        return redirect('index')

    else:

        messages.error(request,'Email or Password is not valid')

        return redirect('login')

else:

    messages.error(request,'Email or Password is not valid')

    return redirect('login')


def doLogout(request):

    logout(request)

    return redirect('login')


@login_required(login_url='/')

def INDEX(request):

    computer_count = AddComputer.objects.all().count()

    user_count = AddCyberUser.objects.filter().count()

    context = {

        'computer_count':computer_count,

        'user_count':user_count,

    }

    return render(request,'index.html',context)
```

```
login_required(login_url='/')
```

```
def PROFILE(request):
```

```
    user = CustomUser.objects.get(id = request.user.id)
```

```
    context = {
```

```
        "user":user,
```

```
    }
```

```
    return render(request,'profile.html',context)
```

```
@login_required(login_url = '/')
```

```
def PROFILE_UPDATE(request):
```

```
    if request.method == "POST":
```

```
        profile_pic = request.FILES.get('profile_pic')
```

```
        first_name = request.POST.get('first_name')
```

```
        last_name = request.POST.get('last_name')
```

```
        email = request.POST.get('email')
```

```
        username = request.POST.get('username')
```

```
        try:
```

```
            customuser = CustomUser.objects.get(id = request.user.id)
```

```
            customuser.first_name = first_name
```

```
            customuser.last_name = last_name
```

```
            customuser.save()
```

```
            messages.success(request,"Your profile has been updated successfully")
```

```
            return redirect('profile')
```

```
        except:
```

```
        messages.error(request,"Your profile updation has been failed")

    return render(request, 'profile.html')


def CHANGE_PASSWORD(request):

    context = {}

    ch = User.objects.filter(id = request.user.id)

    if len(ch)>0:

        data = User.objects.get(id = request.user.id)

        context["data"]:data

    if request.method == "POST":

        current = request.POST["cpwd"]

        new_pas = request.POST['npwd']

        user = User.objects.get(id = request.user.id)

        un = user.username

        check = user.check_password(current)

        if check == True:

            user.set_password(new_pas)

            user.save()

            messages.success(request,'Password Change  Succesfully!!!')

            user = User.objects.get(username=un)

            login(request,user)

        else:

            messages.success(request,'Current Password wrong!!!')

            return redirect("change_password")
```

```
return render(request,'change-password.html')
```

```
@login_required(login_url='/')
```

```
def Add_Computer(request):
```

```
    if request.method == "POST":
```

```
        compname = request.POST.get('compname')
```

```
        comploc = request.POST.get('comploc')
```

```
        idadd = request.POST.get('idadd')
```

```
        computer = AddComputer(compname = compname,
```

```
                                comploc = comploc,
```

```
                                idadd = idadd)
```

```
        computer.save()
```

```
        messages.success(request,"computer details has been saved")
```

```
        return redirect('add_computer')
```

```
    return render(request, 'add-computer.html')
```

```
login_required(login_url='/')
```

```
def MANAGE_Computer(request):
```

```
    computer = AddComputer.objects.all()
```

```
    context = {
```

```
        'computer':computer,
```

```
    }
```

```
    return render(request,'manage-computer.html',context)
```



```
def DELETE_Computer(request,id):

    computer = AddComputer.objects.get(id=id)

    computer.delete()

    messages.success(request,'Record Delete Succesfully!!!')

    return redirect('manage_computer')

    login_required(login_url='/')

def UPDATE_Computer(request,id):

    computer = AddComputer.objects.get(id=id)

    context = {

        'computer':computer,

    }

    return render(request,'update-computer.html',context)


def UPDATE_COMPUTER_DETAILS(request):

    if request.method == 'POST':

        computer_id = request.POST.get('computer_id')

        compname = request.POST['compname']

        comploc = request.POST['comploc']

        idadd = request.POST['idadd']

        computer =AddComputer.objects.get(id=computer_id)

        computer.compname = compname

        computer.comploc = comploc

        computer.idadd = idadd

        computer.save()
```

```
messages.success(request,"Your computer detail has been updated successfully")

return redirect('manage_computer')

return render(request, 'update_computer.html')


@login_required(login_url = '/')

def Add_User(request):

    computer = AddComputer.objects.all()

    if request.method == "POST":

        name = request.POST.get('name')

        address = request.POST.get('address')

        mobilenumber = request.POST.get('mobilenumber')

        email = request.POST.get('email')

        idproof = request.POST.get('idproof')

        computeruse_id = request.POST.get('computeruse_id')

        entryid = request.POST.get('entryid')

        if AddCyberUser.objects.filter(entryid=entryid).exists():

            messages.success(request,"This entry id is already exist")

            return redirect('add_user')

        else:

            computer =AddComputer.objects.get(id=computeruse_id)

            userdetail = AddCyberUser(name = name,

            address = address,

            mobilenumber = mobilenumber,

            email=email,
```

```
        idproof=idproof,

        entryid=entryid,

        computeruse_id=computer

    )

    userdetail.save()

    messages.success(request,"User details has been saved")

    return redirect('add_user')

context = {'computer':computer}

return render(request, 'add-cyber-user.html',context)


login_required(login_url='/')

def MANAGE_User(request):

    cyber_user = AddCyberUser.objects.all()

    context = {

        'cyber_user':cyber_user,

    }

    return render(request,'manage-cyber-user.html',context)


def DELETE_User(request,id):

    user = AddCyberUser.objects.get(id=id)

    user.delete()

    messages.success(request,'Record Delete Succesfully!!!')

    return redirect('manage_user')

login_required(login_url='/')
```

```
def UPDATE_USER(request,id):

    user = AddCyberUser.objects.filter(id=id)

    context = {

        'user':user,

    }

    return render(request,'update-cyber-user.html',context)


def UPDATE_REMARK(request):

    if request.method == 'POST':

        user_id= request.POST.get('user_id')

        remark = request.POST['remark']

        status = request.POST['status']

        fees = request.POST['fees']

        cyberuser= AddCyberUser.objects.get(id=user_id)

        cyberuser.remark = remark

        cyberuser.status = status

        cyberuser.fees = fees

        cyberuser.save()

        messages.success(request,"Remark has been updated successfully")

        return redirect('manage_user')

    return render(request,'update-visitor.html')


login_required(login_url='/')

def Manage_Old_USER(request):
```

```
cyber_user = AddCyberUser.objects.all()

context = {

    'cyber_user':cyber_user,

}

return render(request,'manage-cyber-old_user.html',context)

login_required(login_url='/')

def VIEW_Old_USER(request,id):

    user = AddCyberUser.objects.filter(id=id)

    context = {

        'user':user,

    }

    return render(request,'view-user_details.html',context)


def Search(request):

    if request.method == "GET":

        query = request.GET.get('query',"")

        if query:

            cyberuser = AddCyberUser.objects.filter(entryid__icontains = query)

            messages.success(request, "Search against this " + query)

            return render(request,'search.html',{'cyberuser':cyberuser})

        else:

            print("No Record Found")

            return render(request,'search.html',{})
```

```
def Between_Date_Report(request):

    if request.method == "POST":

        fdate = request.POST.get('fdate')

        todate = request.POST.get('todate')

        searchresult = AddCyberUser.objects.filter(created_at__gte=fdate,created_at__lte=todate)

        messages.success(request, "Search data from " + fdate + "to" + todate)

        return render(request,'between-date.html',{'data':searchresult})

    else:

        print("No Record Found")

        return render(request,'between-date.html',{})


def TotalUser(request):

    cyber_user = AddCyberUser.objects.all()

    context = {

        'cyber_user': cyber_user,

    }

    return render(request, 'total-users.html', context)
```

3.4.2 Urls.py:

```
from django.contrib import admin

from django.urls import path

from django.conf import settings

from django.conf.urls.static import static

from . import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('base/', views.BASE, name='base'),

    path("", views.LOGIN, name='login'),

    path('doLogin', views.doLogin, name='doLogin'),

    path('doLogout', views.doLogout, name='logout'),

    path('Index', views.INDEX, name='index'),

    #profile path

    path('Profile', views.PROFILE, name='profile'),

    path('Profile/update', views.PROFILE_UPDATE, name='profile_update'),

    path('Password', views.CHANGE_PASSWORD, name='change_password'),

    path('AddComputer', views.Add_Computer, name='add_computer'),

    path('ManageComputer', views.MANAGE_Computer, name='manage_computer'),

    path('DeleteComputer/<str:id>', views.DELETE_Computer, name='delete_computer'),

    path('UpdateComputer/<str:id>', views.UPDATE_Computer, name='update_computer'),

    path('UPDATE_COMPUTER_DETAILS', views.UPDATE_COMPUTER_DETAILS,
name='update_computer_details'),
```

#cyber user

```
path('AddUser', views.Add_User, name='add_user'),  
path('ManageCyberUser', views.MANAGE_User, name='manage_user'),  
path('DeleteUser/<str:id>', views.DELETE_User, name='delete_user'),  
path('UpdateCyberUser/<str:id>', views.UPDATE_USER, name='update_user'),  
path('UpdateRemark/Update', views.UPDATE_REMARK, name='update_remark'),  
path('ManageOldUser', views.Manage_Old_USER, name='manage_old_user'),  
path('TotalUsers', views.TotalUser, name='total_users'),  
path('ViewOldUser/<str:id>', views.VIEW_Old_USER, name='view_user_details'),  
path('Search', views.Search, name='search'),  
path('BetweenDateReport', views.Between_Date_Report, name='between_date_report')
```

```
] + static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
```


Chapter 4

RESULTS

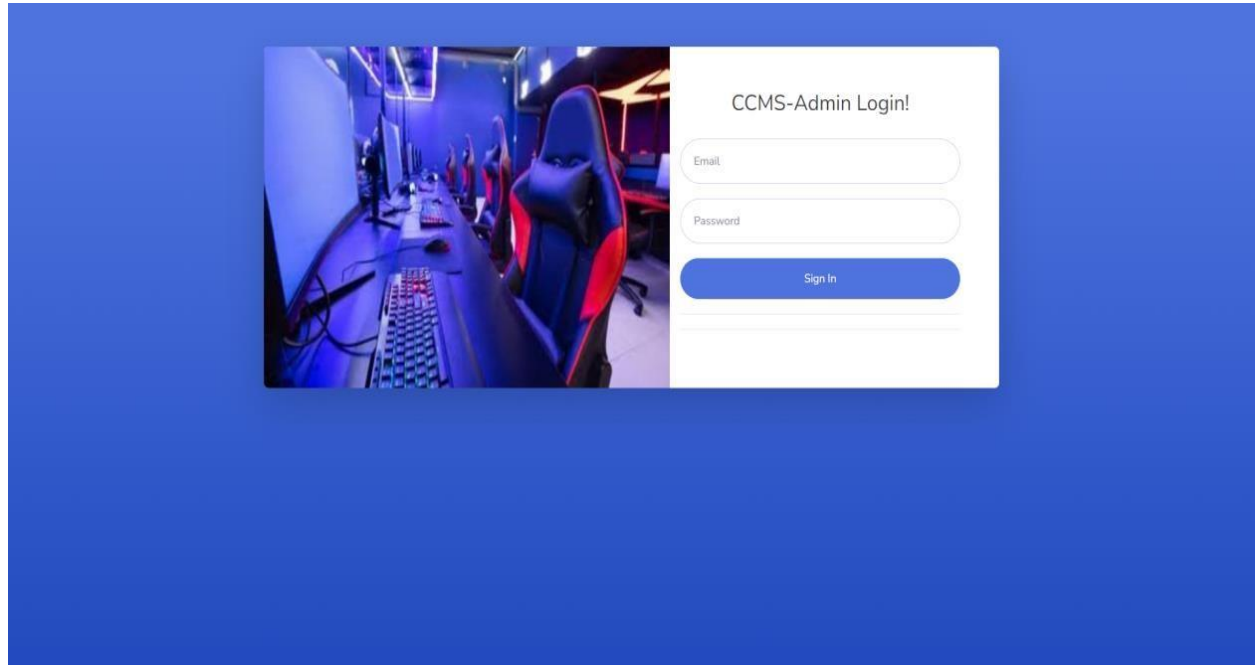


Fig 4.1 Admin Login

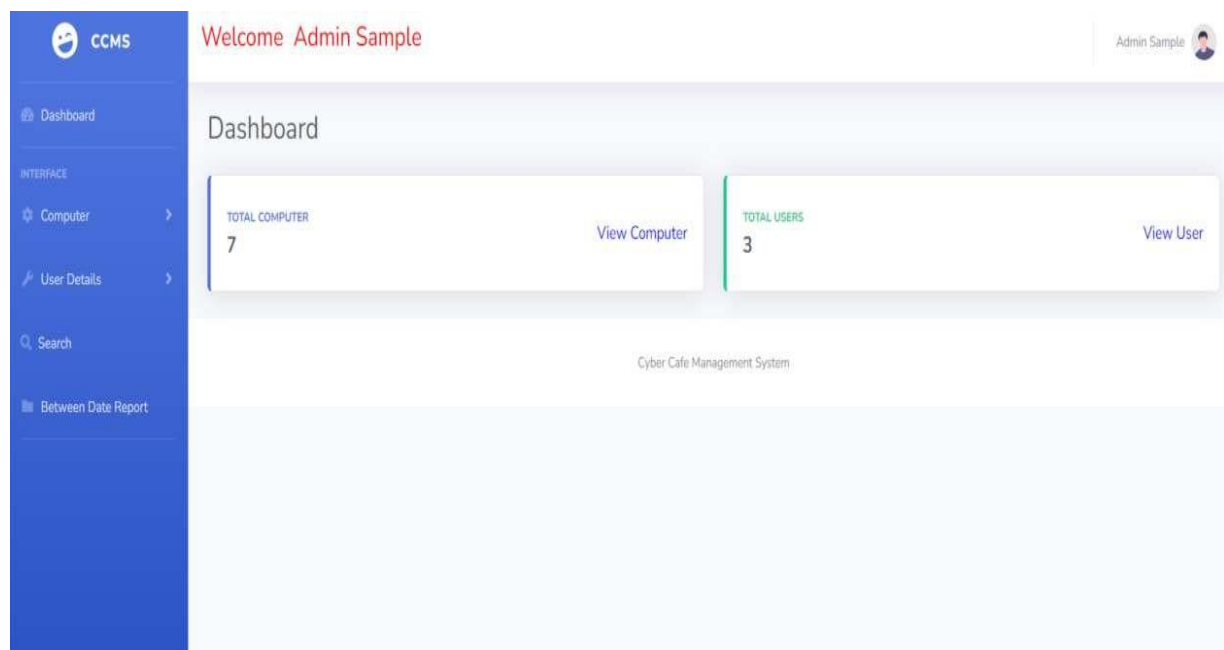


Fig 4.2 Dashboard

The screenshot shows the 'Admin Profile' page in the CCMS interface. The left sidebar contains the CCMS logo and navigation links: Dashboard, Computer, User Details, Search, and Between Date Report. The main content area has a header 'Welcome Admin Sample' and a user profile icon. The 'Admin Profile' section contains a form with the following fields: First Name (Admin), Last Name (Sample), Email (admin@gmail.com), and Username (admin). A blue 'Update' button is at the bottom of the form. The footer text is 'Cyber Cafe Management System'.

Fig 4.3 Admin Profile

The screenshot shows the 'Change Password' page in the CCMS interface. The left sidebar is identical to the previous figure. The main content area has a header 'Welcome Admin Sample' and a user profile icon. The 'Change Password' section contains a form with two fields: Current Password and New Password. A blue 'Change' button is at the bottom of the form. The footer text is 'Cyber Cafe Management System'.

Fig 4.4 Change Password

CCMS

Welcome Admin Sample

Admin Sample

Dashboard

INTERFACE

Computer

User Details

Search

Between Date Report

Add Computer

Computer Name

Computer Locations

ID Address

Add

Cyber Cafe Management System

Fig 4.5 Add Computer

CCMS

Welcome Admin Sample

Admin Sample

Dashboard

INTERFACE

Computer

User Details

Search

Between Date Report

Manage Computer Details

S.No	Name of Computer	Location of Computer	IP Address	Creation Date	Action
1	Acer	Cabin101	127.0.0.06	Dec. 18, 2023, 6:39 a.m.	<button>Update</button> <button>Delete</button>
2	DELL	Cabin102	127.0.0.08	Dec. 18, 2023, 6:39 a.m.	<button>Update</button> <button>Delete</button>
3	Asus Gaming Laptop	Cabin103	127.0.0.2	Dec. 18, 2023, 6:39 a.m.	<button>Update</button> <button>Delete</button>
4	Dell	Cabin104	127.0.0.5	Dec. 18, 2023, 6:39 a.m.	<button>Update</button> <button>Delete</button>
5	Asus Gaming Laptop	Cabin109	127.0.0.09	Dec. 18, 2023, 6:39 a.m.	<button>Update</button> <button>Delete</button>
6	ASUS	Cabin109	127.0.0.9	Dec. 18, 2023, 6:40 a.m.	<button>Update</button> <button>Delete</button>
7	Dell	Cabin1010	127.0.0.10	Dec. 18, 2023, 6:40 a.m.	<button>Update</button> <button>Delete</button>
S.No	Name of Computer	Location of Computer	IP Address	Creation Date	Action

Cyber Cafe Management System

Fig 4.6 Manage Computer

The screenshot shows the 'Update Computer' form within the CCMS interface. The left sidebar contains navigation links: Dashboard, Computer, User Details, Search, and Between Date Report. The top header displays 'Welcome Admin Sample' and the user's name 'Admin Sample'. The form itself has three input fields: 'Computer Name' with the value 'Acer', 'Computer Locations' with 'Cabin101', and 'ID Address' with '127.0.0.06'. A blue 'Update' button is positioned at the bottom of the form. The footer of the page reads 'Cyber Cafe Management System'.

Fig 4.7 Update Computer

The screenshot displays the 'Add Cyber User' form in the CCMS interface. The layout is consistent with the previous figure, featuring the same sidebar and header. The form includes six text input fields: 'User Name', 'User Address', 'Mobile Number', 'Email', 'ID Proof', and 'Entry ID'. Below these is a dropdown menu labeled 'Select Computer' with the option 'Choose Computer' selected. A blue 'Add' button is located at the bottom of the form. The footer text 'Cyber Cafe Management System' is visible at the bottom of the page.

Fig 4.8 Add Users

Welcome Admin Sample

Manage New Users Details

S.No	Entry ID	Name of User	Email	Mobile Number	Cabin User	Entry Date/Entry Time	Status	Checkout Date/Checkout Time	Action
2	1232	Shanu Mishra	shanu@gmail.com	2147483647	Asus Gaming Laptop(Cabin109)	Dec. 15, 2023, 6:42 a.m.	Not Updated Yet	Dec. 18, 2023, 6:42 a.m.	Update Delete
3	123	Sarita	sar@gmail.com	2147483647	Asus Gaming Laptop(Cabin103)	Dec. 18, 2023, 6:43 a.m.	Not Updated Yet	Dec. 18, 2023, 6:43 a.m.	Update Delete

Cyber Cafe Management System

Fig 4.9 Manage New User Details

Welcome

Update Cyber User

Entry ID	1232
Full Name	Shanu Mishra
Email	shanu@gmail.com
Mobile Number	2147483647
Address	O-901, GHU, Block-8
ID Proof Number	77797jkh
Computer Used	Asus Gaming Laptop
Entry Date/Entering Time	Dec. 15, 2023, 6:42 a.m.
Remark	Not Updatet Yet
Staus	Not Updatet Yet
Out Time	Dec. 18, 2023, 6:42 a.m.

Remark

Status

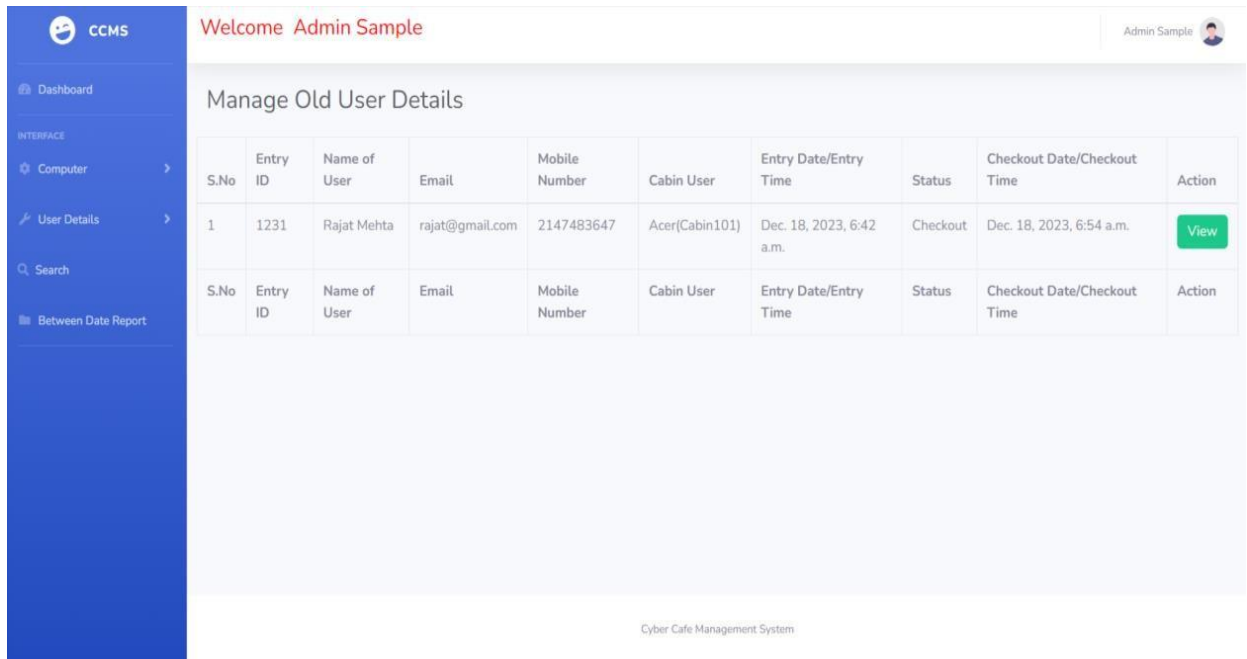
Checkout

Fees

[Update](#)

Cyber Cafe Management System

Fig 4.10 Update New Users Details



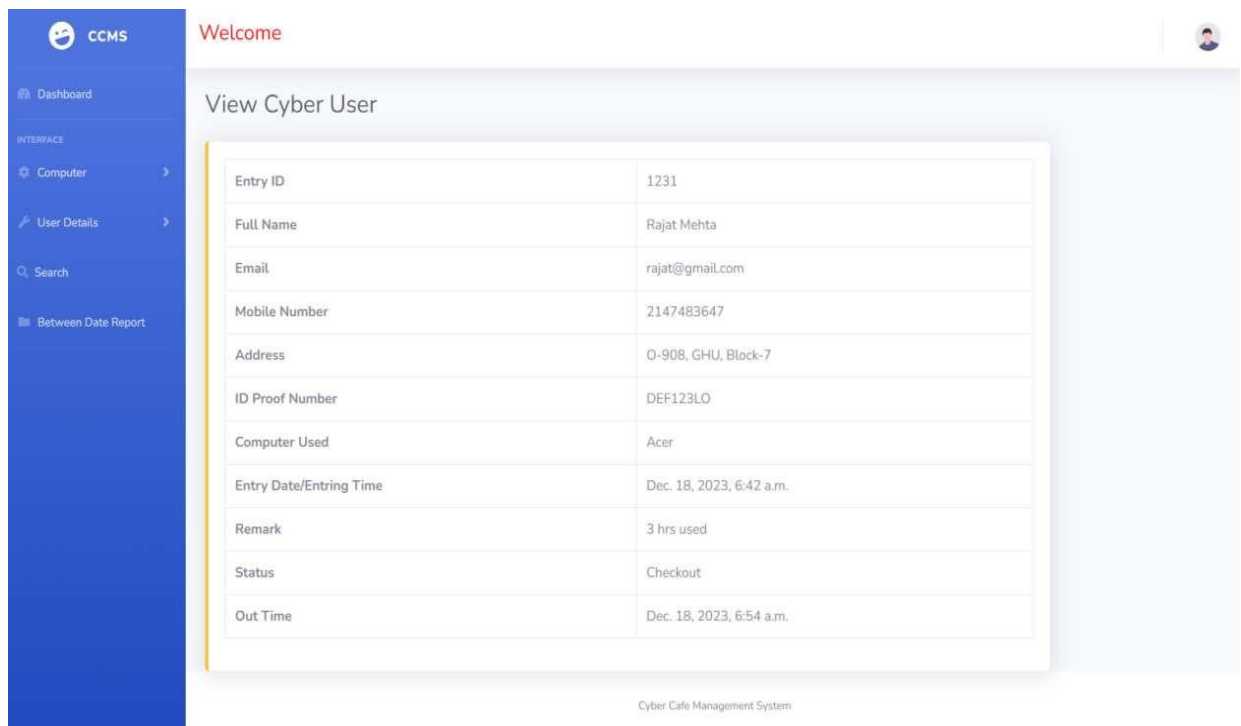
Welcome Admin Sample

Manage Old User Details

S.No	Entry ID	Name of User	Email	Mobile Number	Cabin User	Entry Date/Entry Time	Status	Checkout Date/Checkout Time	Action
1	1231	Rajat Mehta	rajat@gmail.com	2147483647	Acer(Cabin101)	Dec. 18, 2023, 6:42 a.m.	Checkout	Dec. 18, 2023, 6:54 a.m.	View

Cyber Cafe Management System

Fig 4.11 Manage Old Users Details



Welcome

View Cyber User

Entry ID	1231
Full Name	Rajat Mehta
Email	rajat@gmail.com
Mobile Number	2147483647
Address	O-908, GHU, Block-7
ID Proof Number	DEF123LO
Computer Used	Acer
Entry Date/Entry Time	Dec. 18, 2023, 6:42 a.m.
Remark	3 hrs used
Status	Checkout
Out Time	Dec. 18, 2023, 6:54 a.m.

Cyber Cafe Management System

Fig 4.12 View Old Users Details

Welcome Admin Sample

Search User Details

Search (By Entry ID)

Search

Search against this 12

S.No	Entry ID	Name of User	Email	Mobile Number	Cabin User	Entry Date/Entry Time	Status	Checkout Date/Checkout Time	Action
1	1231	Rajat Mehta	rajat@gmail.com	2147483647	Acer	Dec. 18, 2023, 6:42 a.m.	Checkout	Dec. 18, 2023, 6:54 a.m.	View
2	1232	Shanu Mishra	shanu@gmail.com	2147483647	Asus Gaming Laptop	Dec. 15, 2023, 6:42 a.m.	Not Updated Yet	Dec. 18, 2023, 6:42 a.m.	View
3	123	Sarita	sar@gmail.com	2147483647	Asus Gaming Laptop	Dec. 18, 2023, 6:43 a.m.	Not Updated Yet	Dec. 18, 2023, 6:43 a.m.	View

Cyber Cafe Management System

Fig 4.13 Search Data

Welcome Admin Sample

Between Dates Reports

From Date
mm/dd/yyyy

To Date
mm/dd/yyyy

Submit

Search data from 2023-12-01 to 2023-12-20

S.No	Entry ID	Name of User	Email	Mobile Number	Cabin User	Entry Date/Entry Time	Status	Checkout Date/Checkout Time	Action
1	1231	Rajat Mehta	rajat@gmail.com	2147483647	Acer	Dec. 18, 2023, 6:42 a.m.	Checkout	Dec. 18, 2023, 6:54 a.m.	View
2	1232	Shanu Mishra	shanu@gmail.com	2147483647	Asus Gaming Laptop	Dec. 15, 2023, 6:42 a.m.	Not Updated Yet	Dec. 18, 2023, 6:42 a.m.	View
3	123	Sarita	sar@gmail.com	2147483647	Asus Gaming Laptop	Dec. 18, 2023, 6:43 a.m.	Not Updated Yet	Dec. 18, 2023, 6:43 a.m.	View

Cyber Cafe Management System

Fig 4.14 Between Reports

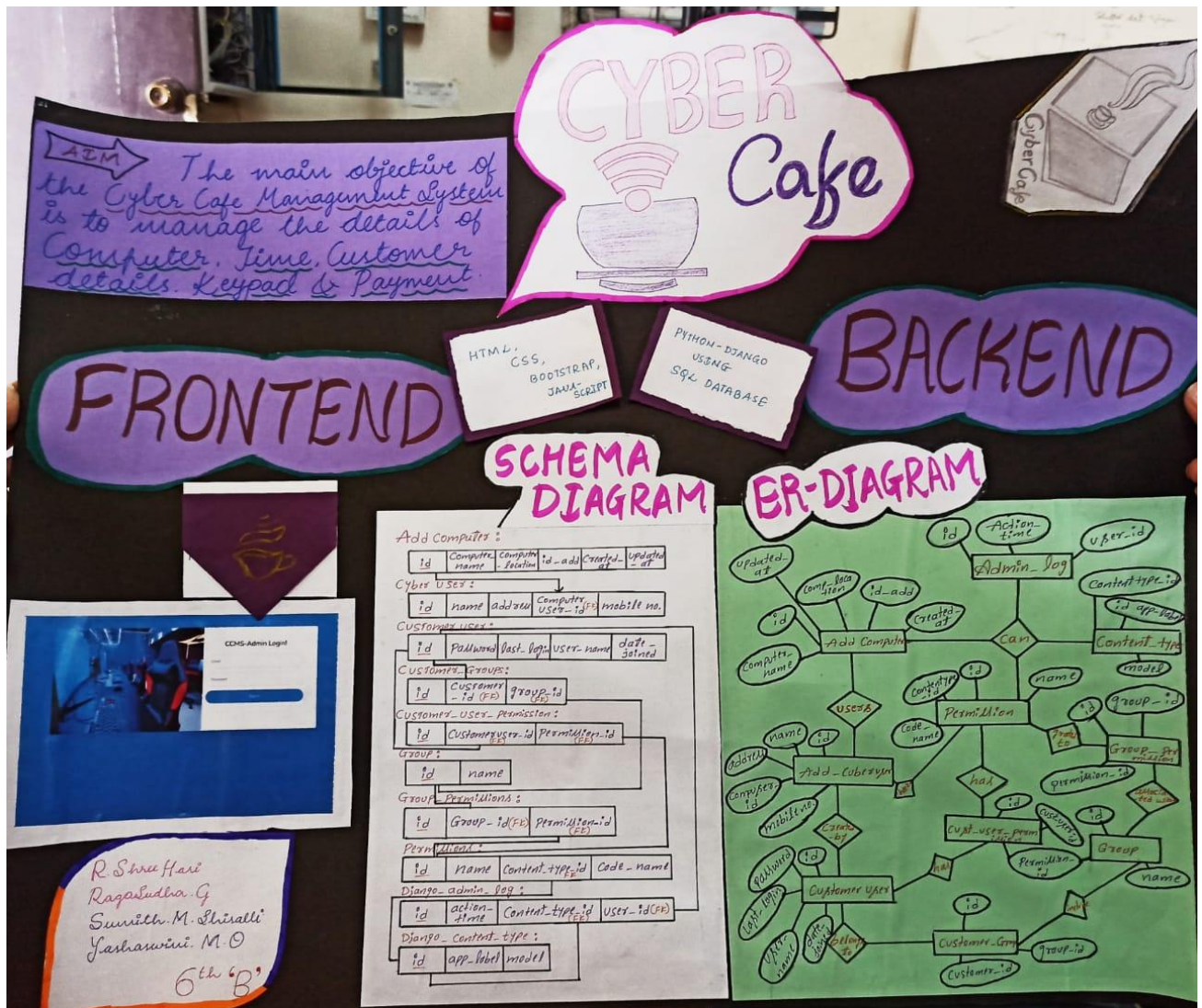


Fig 4.15 Snapshot of Chart Prepared

Chapter 5

CONCLUSION

The Cyber Cafe Management System developed using Django provides an efficient and streamlined solution for managing the day-to-day operations of a cyber cafe. By leveraging Django's robust framework, the system ensures secure and scalable management of user accounts, session monitoring, and payment processing. Key features include real-time tracking of computer usage, automated billing, and detailed reporting, which facilitate seamless operations and enhance customer satisfaction. Additionally, the system's intuitive interface allows for easy navigation and management, reducing the administrative burden on staff. Overall, this Django-based management system significantly improves operational efficiency, ensuring a smooth and enjoyable experience for both customers and cafe operators.

Our software incorporates all the features and facilities provided by the Visual Studio software. This project has been developed to manage the entire working of the Cyber Café. Our software simplifies and replaces all the manual effort and the paper works done by the owner of the cyber to a completely electronically environment, be it bill generation or customer creation and fulfilment of their needs and customer satisfaction. Hence both the customer and the owner are at their ease.

BIBLIOGRAPHY

For Python and Django

- [1] Django homepage. <http://www.djangoproject.com/>.
- [2] Python documentation. <http://www.python.org/doc>.
- [3] Django (web framework). <http://en.wikipedia.org/wiki/Django>.
- [4] Django documentation. <http://docs.djangoproject.com>.
- [5] Python (programming language). <http://en.wikipedia.org/wiki/Python>
- [6] Books: Web Development with Django by Ben Shaw, Saurabh, Django 4 by Antonio Mele.

For MySQL

- <https://www.mysql.com/>
- <http://www.mysqltutorial.org>

For XAMPP

- <https://www.apachefriends.org/download.html>