

Voice of Charity: Prospecting the Donation Recurrence & Donor Retention in Crowdfund

Submitted in partial fulfillment of the requirements for the award of
Bachelor Degree in Computer Science

By

S. JENIFER (Reg. No. 39290041)

A. JENITHA (Reg. No. 39290042)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

MARCH 2022



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **S. JENIFER (39290041)** and **A. JENITHA (39290042)** who carried out the project entitled **"Voice of Charity: Prospecting the Donation Recurrence & Donor Retention in Crowd funding"** under my supervision from _____ to _____.

Internal Guide

DR.G. KALAIARASI

Head of the Department

Dr. L. LAKSHMANAN M.E., Ph.D. and Dr.S.Vigneshwari M.E., Ph.D.

Internal Examiner

External Examiner

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I, **S. JENIFER (Reg. No. 39290041)** and **A. JENITHA (Reg. No. 39290042)** hereby declare that the Project Report entitled " **Voice of Charity: Prospecting the Donation Recurrence & Donor Retention in Crowd funding**" done by me under the guidance of **Dr. KALAIARASI** is submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing and **Dr. S. VIGNESHWARI, M.E., Ph.D.**, and **Dr.S.Vigneshwari M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. KALAIARASI, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

ABSTRACT

Crowdfunding is a practice of raising funds from people to support your project which has brought new life to charity, i.e., making it easy to donate any amount of money to help across the globe. Donation-based crowdfunding is the most preferred mode of fundraising. Crowdfunding through online platforms knows no boundaries, and has the potential to go viral. The problem of high donor attrition i.e., many donors donate only once or very few times within a rather short lifecycle and then leave. Thus, it is an urgent task to analyse the factors of and then further predict the donor behaviour. In this paper, we present a focused study on analysis of donation recurrence and donor retention to predict the donor's interest in donation. Specifically, we proposed a model, which has the details of recipient, actual donor and the verifying person. After the donation process, every donor will get a proper donation certificate approved by the government. The experimental results will clearly demonstrate the individual's interest for donation and to appreciate them to donate more in their future with a proper secured transaction with the support of the government.

TABLE OF CONTENTS

CHAPTER No	TITLE	PAGE
	ABSTRACT	VI
	LIST OF FIGURES	IX
1	INTRODUCTION	1
	1.1 ABOUT THE PROJECT	1
2	SYSTEM ANALYSIS	2
	2.1 EXISTING SYSTEM	2
	2.2 PROPOSED SYSTEM	2
3	REQUIREMENTS SPECIFICATION	3
	3.1 INTRODUCTION	3
	3.2 HARDWARE AND SOFTWARE SPECIFICATION	5
	3.3 TECHNOLOGIES USED	5
	3.3.1 JAVA	5
	3.3.3.1 INTRODUCTION TO JAVA	5
	3.3.1.2 WORKING OF JAVA	7
4	PROJECT PURPOSE AND SCOPE	15
	4.1 PURPOSE	15
	4.2 PROJECT SCOPE	15
	4.3 PRODUCT PERSPECTIVE	16
	4.4 SYSTEM FEATURES	16
	4.5 DESIGN AND IMPLEMENTATION CONSTRAINTS	17
	4.6 OTHER NONFUNCTIONAL REQUIREMENTS	18
5	SYSTEM DESIGN	20
	5.1 ARCHITECTURE DIAGRAM	20

6	SYSTEM DESIGN – DETAILED	28
	6.1 MODULES	29
	6.2 MODULE EXPLANATION	29
7	CODING AND TESTING	30
	7.1 CODING	30
	7.2 CODING STANDARDS	30
	7.3 TEST PROCEDURE	33
	7.4 TESTING CODE	34
8	RESULT AND DISCUSION	39
	8.1 HOME PAGE	39
	8.2 RECIPIENT REGISTER	40
	8.3 ROVIDING FUND FROM DONOR TO RECIPIENT	41
9	CONCLUSION AND FUTURE WORK	41
	REFERENCES	42
	SOURCE CODE	46

LIST OF FIGURES

FIGURE	FIGURE NAME	PAGE
NO		NO
5.1	SYSTEM DESIGN	20
5.2	SEQUENCE DIAGRAM	21
5.3	USE CASE DIAGRAM	23
5.4	ACTIVITY DIAGRAM	24
5.5	COLLABORATION DIAGRAM	25
5.6	DATA FLOW DIAGRAM	26,27
5.7	CLASS DIAGRAM	28
8.1	HOME PAGE	39
8.2	RECIPIENT REGISTER	40
8.3	PROVIDING FUND FROM FROM DONOR TO RECIPIENT	41

CHAPTER 1

INTRODUCTION

Aim:

The main aim of this project is to clearly demonstrate the effectiveness for analyzing and predicting the donation recurrence and donor retention in crowdfunding and providing a proper trustworthy donation from the donors to the clients.

Synopsis:

The emergence of mobile cloud computing enables mobile users to offload applications to nearby mobile resource to reduce energy consumption and improve performance. However due to mobility and cloudlet capacity the connections between a mobile user and mobile cloudlets can be mobile services. As a result, offloading actions taken by the mobile user may fail (e.g., the user moves out of communication range of cloudlets). The mobile user has an application to be executed. In the intermediate mobile act as mobile provider to perform user task with neighbours offload node. As the application is divided into code sections during the execution the mobile user can dynamically decide to execute application phases locally on the mobile device or offload to nearby cloudlet mobiles.

The incentive scheme uses a virtual currency, named FlopCoin, to compensate devices whenever they execute an offloadable task. The amount of the FlopCoins that are exchanged in each application offloading is determined by individually rational and incentive methods.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In the existing system, donation recurrence and donor retention was predicted with the help of large-scale behavioral data collected from crowdfunding. It uses the Joint Deep Survival model to integrate the heterogeneous data since they are highly relevant. This system clearly demonstrates the effectiveness for analyzing and predicting the donation recurrence and donor retention in crowdfunding. The retention of the donor is not clearly known because they have not observed the occurrence of donor attrition. The models may lose the abilities of capturing the sequence dependence for such a long time.

PROBLEM DEFINITION

- The donor and the recipient are not authenticated so we cannot identify whether they are verified authenticated users.
- The donor may not know whether the recipient is getting the whole amount of money paid by the donor.
- Malpractice may occur in the name of donation in the income tax department by some donors.

2.2 PROPOSED SYSTEM

In this work, the client and the donor have to fill their personal details which will be verified by the third party, the verifying agent appointed by the government. The verifying agent will accept the details by verifying their details and proceeds the secured transaction from the donors to the clients. This will make the secure transaction only after the secured authentication. So that only the authenticated users and donors can donate or ask for donation. This may avoid the income tax frauds or those who escape from tax pay by false statements of money lending.

ADVANTAGES:

- By analyzing and predicting the donation recurrence and donor retention, the donors can be predicted easily and are contacted for any need of money from the recipient.
- The donation recurrence and donor retention are highly correlated and indicates the effectiveness of JDS structure.

CHAPTER 3**REQUIREMENT SPECIFICATIONS****3.1 INTRODUCTION**

CROWDFUNDING is an emerging Internet-based fundraising mechanism soliciting small monetary contributions from crowd donors to help others in trouble or with dreams [1]. Recent years have witnessed the rapid development of crowdfunding platforms among which the donation-based ones are becoming increasingly popular [1], [2], such as Kiva.org 1 [3], and DonorsChoose.org 2 [4]. Leveraging Internet, crowdfunding has brought new life to charity, i.e., making it easy to donate any amount of money even every penny to help others across the globe.

For example, Kiva.org is an international nonprofit platform, founded in 2005, with a mission to connect people through lending to alleviate poverty. Specifically, Kiva.org enables Field Partners (nonprofit organizations around the world) to screen the needy or suffering, and post requests in the form of projects to Kiva.org for funding. Then the accessing donors crowdfund these projects in increments of \$25 or more. Donors may act as individuals or teams. The critical component for the success of crowdfunding communities is the recruitment and continued engagement of donors [4]. However, because of the non-profit nature, the situation relating to

donor retention for donation-based crowdfunding as well as traditional charities is extremely serious, i.e., usually, the donor attrition rate is above 70% [4]. Actually, customer attrition/churn [5], [6] is crucial and highly focused on in many commercial scenarios, such as E-commerce, finance and services. However, for a quite long time, relevant studies on analyzing donor retention in charity have been rather limited in the literature. Fortunately, with the accumulation of large-scale user behavior data in crowdfunding platforms, many data-driven studies which focus on analyzing the user behaviors have been conducted [7], [8]. For example, Liu, et al. [7] studied the donation motivation classification in Kiva.com. Especially, Althoff, et al. [4] explored various factors impacting donor retention in DonorsChoose.org from the statistical perspectives which was inspiring for our research. However, how to comprehensively analyze the heterogeneous factors affecting and then further predict the donor retention or attrition, are still largely unexplored areas, both in the charity and in other domains. In addition to these heterogeneous factors, according to our observation and analysis, donors' own behaviors (i.e., donation recurrence) could particularly reflect their decision on retention. In fact, donation recurrence prediction is an inevitable intermediate goal for predicting the donor retention. Thus, in this paper, we attempt to track this problem by jointly predicting the donor retention and also the intermediate goal (predicting donation recurrence). Although it is necessary to construct the predictions of donation recurrence and donor retention, as they can alert platforms that they need to do something before they lose donors, this is a very challenging task. First, donor behaviors, e.g., donation recurrence, donor retention or attrition, are influenced by various factors [4], such as their motives and preferences, their social contacts in crowdfunding communities, and the characteristics of the projects to which they have recently donated. How to comprehensively analyse the heterogeneous features and integrate them for accurate prediction is not a trivial issue. Second, according to our data analysis, the behaviours of donors, especially the donation recurrence, are highly correlated with their retention or attrition. How to model the relations of donation recurrence and donor retention and further synchronously predict these two behavioral events with a joint model are quite open problems. Finally, the presence of a large amount of censored data [9], [10], i.e., the exact attrition outcomes of some donors are unobservable or they

do not perform any behaviours (donation or attrition) during our monitoring periods, imposes significant challenges in relation to this problem. Because many donors may be still in the platform in our data and most lost donors do not explicitly close their accounts when leaving, the censoring phenomenon is an inescapable concern.

3.2 HARDWARE AND SOFTWARE SPECIFICATION

3.2.1 HARDWARE REQUIREMENTS

- Hard Disk : 80GB and Above
- RAM : 4GB and Above
- Processor : P IV and Above

3.2.2 SOFTWARE REQUIREMENTS

- Windows 7 and above
- JDK 1.7
- Tomcat 7.0
- MySQL

3.3 TECHNOLOGIES USED

- J2EE (JSP, Servlets), JavaScript, HTML, CSS.

3.3.1 JAVA

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

3.3.3.1 INTRODUCTION TO JAVA

Java has been around since 1991, developed by a small team of Sun Microsystems developers in a project originally called the Green project. The intent of the project was to develop a platform-independent software technology that would be used in the consumer electronics industry. The language that the team created was originally called Oak.

The first implementation of Oak was in a PDA-type device called Star Seven (*7) that consisted of the Oak language, an operating system called GreenOS, a user interface, and hardware. The name *7 was derived from the telephone sequence that was used in the team's office and that was dialed in order to answer any ringing telephone from any other phone in the office.

Around the time the FirstPerson project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The World Wide Web, a name applied to the Internet's millions of linked HTML documents was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical Web browser called Mosaic, developed by ncSA. The browser simplified Web browsing by combining text and graphics into a single interface to eliminate the need for users to learn many confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called Java. Soon after, the Java compiler was rewritten in the Java language from its original C code, thus proving that Java could be used effectively as an application language. Sun introduced Java in May 1995 at the Sun World 95 convention.

Web surfing has become an enormously popular practice among millions of computer users. Until Java, however, the content of information on the Internet has been a bland series of HTML documents. Web users are hungry for applications that are interactive, that users can execute no matter what hardware or software platform they are using, and that travel across heterogeneous networks and do not spread viruses to their computers. Java can create such applications.

3.3.1.2 WORKING OF JAVA

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it looks for a particular method by the name of main, which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv [] of C), and is declared as a static method.

To output text from the program, we execute the println method of System.out, which is java's output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user's program.

Java consists of two things:

- Programming language
- Platform

3.3.1.3 THE JAVA PROGRAMMING LANGUAGE

Java is a high-level programming language that is all of the following:

- Simple
- Object-oriented
- Distributed
- Interpreted
- Robust
- Secure
- Architecture-neutral
- Portable

- High-performance
- Multithreaded
- Dynamic

The code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Java is unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes** – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works:

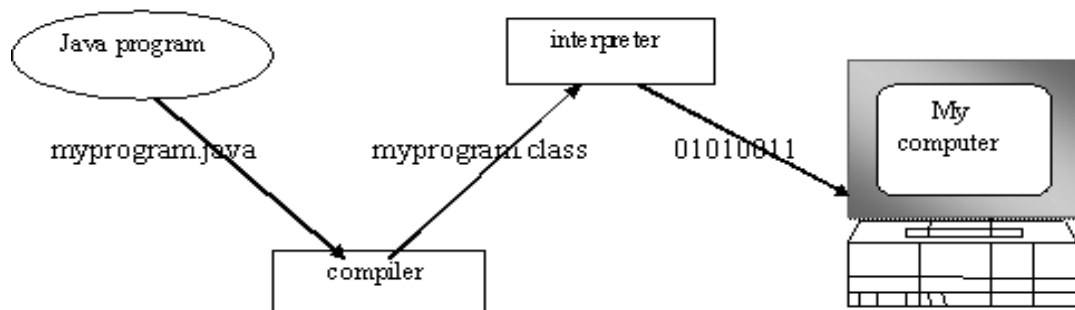


Fig.3.1

You can think of Java byte codes as the machine code instructions for the **Java Virtual Machine (JVM)**. Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of JVM. That JVM can also be implemented in hardware. Java byte codes help make "write once, run anywhere" possible.

You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM.

For example, that same Java program can run on Windows NT, Solaris and Macintos.

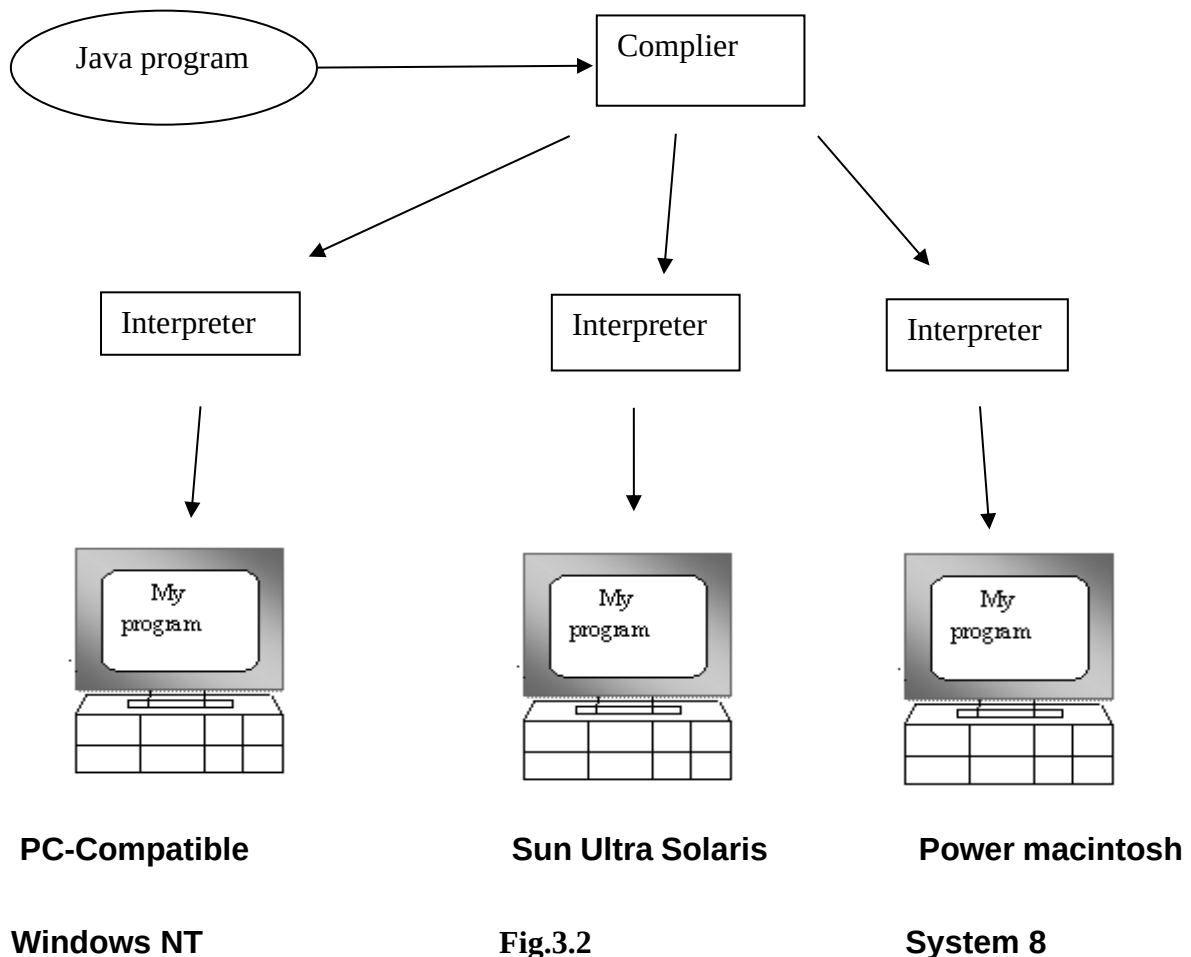


Fig.3.2

3.3.1.4 THE JAVA PLATFORM

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components:

- The Java Virtual Machine (JVM)

- The Java Application Programming Interface (Java API)

You've already been introduced to the JVM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (**packages**) of related components. The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

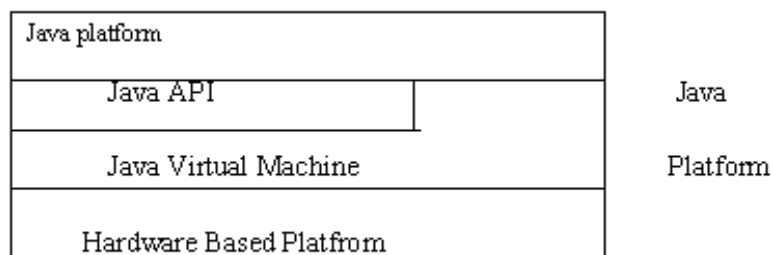


Fig.3.3

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, weel-tuned interpreters, and just-in-time byte compilers can bring Java's performance close to that of native code without threatening portability.

3.3.2 APACHE TOMCAT SERVER

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to

run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

Environment

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

PRODUCT FEATURES

Tomcat 3.x (initial release)

- implements the Servlet 2.2 and JSP 1.1 specifications
- servlet reloading
- basic HTTP functionality Tomcat 4.x
- implements the Servlet 2.3 and JSP 1.2 specifications
- servlet container redesigned as Catalina
- JSP engine redesigned as Jasper
- Coyote connector
- Java Management Extensions (JMX), JSP and Struts-based
- administration
- Tomcat 5.x
- implements the Servlet 2.4 and JSP 2.0 specifications
- reduced garbage collection, improved performance and scalability
- native Windows and Unix wrappers for platform integration

- faster JSP parsing

HISTORY

Tomcat started off as a servlet specification implementation by James Duncan Davidson, a software architect at Sun. He later helped make the project open source and played a key role in its donation by Sun to the Apache Software Foundation.

Davidson had initially hoped that the project would become open-sourced and, since most open-source projects had O'Reilly books associated with them featuring an animal on the cover, he wanted to name the project after an animal. He came up with Tomcat since he reasoned the animal represented something that could take care of and fend for itself. His wish to see an animal cover eventually came true when O'Reilly published their Tomcat book with a tomcat on the cover.

3.3.3 ANDROID INTRODUCTION:

Android is a Linux based operating system it is designed primarily for touch screen mobile devices such as smart phones and tablet computers. The operating system has developed a lot in last 15 years starting from black and white phones to recent smart phones or mini computers. One of the most widely used mobile OS these days is android. The android is software that was founded in Palo Alto of California in 2003.

The android is a powerful operating system and it supports large number of applications in Smartphones. These applications are more comfortable and advanced for the users. The hardware that supports android software is based on ARM architecture platform. The android is an open source operating system means that it's free and any one can use it. The android has got millions of apps available that can help you managing your life one or other way and it is available low cost in market at that reasons android is very popular.

The android development supports with the full java programming language. Even other packages that are API and JSE are not supported. The first version 1.0 of android development kit (SDK) was released in 2008 and latest updated version is jelly bean.

The android is a operating system and is a stack of software components which is divided into five sections and four main layers that is



3.3.3.1 LINUX KERNEL:

The android uses the powerful Linux kernel and it supports wide range of hardware drivers. The kernel is the heart of the operating system that manages input and output requests from software. This provides basic system functionalities like process management, memory management, device management like camera, keypad, display etc the kernel handles all the things. The Linux is really good at networking and it is not necessary to interface it to the peripheral hardware. The kernel itself does not interact directly with the user but rather interacts with the shell and other programs as well as with the hard ware devices on the system.

3.3.3.2 LIBRARIES:

The on top of a Linux kernel there is a set of libraries including open source web browser such as webkit, library libc. These libraries are used to play and record audio and video. The SQLite is a data base which is useful for storage and sharing of application data. The SSL libraries are responsible for internet security etc.

3.3.3.3 ANDROID RUNTIME:

The android runtime provides a key component called Dalvik Virtual Machine which is a kind of java virtual machine. It is specially designed and optimized for android. The Dalvik VM is the process virtual machine in the android operating system. It is software that runs apps on android devices.

The Dalvik VM makes use of Linux core features like memory management and multithreading which is in a java language. The Dalvik VM enables every android application to run its own process. The Dalvik VM executes the files in the dex format.

3.3.3.4 APPLICATION FRAME WORK:

The application framework layer provides many higher level services to applications such as windows manager, view system, package manager, resource manager etc. The application developers are allowed to make use of these services in their application.

3.3.3.5 APPLICATIONS:

You will find all the android applications at the top layer and you will write your application and install on this layer. Examples of such applications are contacts, books, browsers, services etc. Each application performs a different role in the overall applications.

ADVANTAGES:

- Android is Linux based open source operating system, it can be developed by any one
- Easy access to the android apps
- You can replace the battery and mass storage, disk drive and UDB option
- It's supports all Google services
- The operating system is able to inform you of a new SMS and Emails or latest updates.
- It supports Multitasking
- Android phone can also function as a router to share internet
- Its free to customize
- Can install a modified ROM
- Its supports 2D and 3D graphics

CHAPTER 4

4.1 PROJECT PURPOSE AND SCOPE

4.1 PURPOSE

The main aim of this project is to clearly demonstrate the effectiveness for analyzing and predicting the donation recurrence and donor retention in crowdfunding and providing a proper trustworthy donation from the donors to the clients.

4.2 PROJECT SCOPE

In this work, the client and the donor have to fill their personal details which will be verified by the third party, the verifying agent appointed by the government. The verifying agent will accept the details proceeds secured transaction from the donors to the clients. This system uses the clustering algorithm to filter the data from

large scale of datasets and uses K-Nearest neighbor algorithm for clustering, which will cluster the similar data items from large dataset according to the user preference. This system will automatically notifies the donors on any particular day, for example, on their birthday, and appreciate the donors to further improve their sequence of donation. This system will make a secure transaction from the donors to the clients only after verifying whether the donors and the clients are authorized persons. And also this will improve the eagerness of donors for donation.

4.3 PRODUCT PERSPECTIVE

Charity is an organization set up to provide help and raise money for those in need. It is generosity and helpfulness especially toward the needy or suffering. Crowdfunding works through individuals or organizations that invest in (or donate to). Crowdfunding projects in return for a potential profit or reward. Investing this way can be risky, so make sure you know what you're doing. So, we are providing the trustworthy crowdfunding through government, which is the funding of a project by a large number of supporters who each contribute a small amount. Only authenticated recipient and donor can request and donate money here. This system uses the clustering algorithm to filter the data from large scale of datasets and uses K-Nearest neighbor algorithm for clustering the similar data from large dataset. These systems help in automatically notifying the donors according to their interest in donation on any particular day, for example, on their birthday, and appreciate the donors to further improve their sequence of donation.

4.4 SYSTEM FEATURES

Crowd funding is a practice of raising funds from people to support your project which has brought new life to charity, i.e., making it easy to donate any amount of money to help across the globe. Donation-based crowd funding is the most preferred mode of fundraising. Crowd funding through online platforms knows no boundaries, and has the potential to go viral. The problem of high donor attrition i.e., many donors donate only once or very few times within a rather short lifecycle

and then leave. Thus, it is an urgent task to analyze the factors of and then further predict the donor behavior. In this paper, we present a focused study on analysis of donation recurrence and donor retention to predict the donor's interest in donation. Specifically, we proposed a model, which has the details of recipient, actual donor and the verifying person. After the donation process, every donor will get a proper donation certificate approved by the government. The experimental results will clearly demonstrate the individual's interest for donation and to appreciate them to donate more in their future with a proper secured transaction with the support of the government.

4.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

4.5.1 CONSTRAINTS IN ANALYSIS

- Constraints as Informal Text
- Constraints as Operational Restrictions
- Constraints Integrated in Existing Model Concepts
- Constraints as a Separate Concept
- Constraints Implied by the Model Structure

4.5.2 CONSTRAINTS IN DESIGN

- Determination of the Involved Classes
- Determination of the Involved Objects
- Determination of the Involved Actions
- Determination of the Require Clauses
- Global actions and Constraint Realization

4.5.3 CONSTRAINTS IN IMPLEMENTATION

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity

or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

4.6 OTHER NON-FUNCTIONAL REQUIREMENTS

4.6.1 PERFORMANCE REQUIREMENTS

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service;
- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

4.6.2 SAFETY REQUIREMENTS

- The software may be safety-critical. If so, there are issues associated with its integrity level
- The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
- If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
- There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
- If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
- Systems with different requirements for safety levels must be separated.

- Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

CHAPTER 5

5.1 ARCHITECTURE DIAGRAM

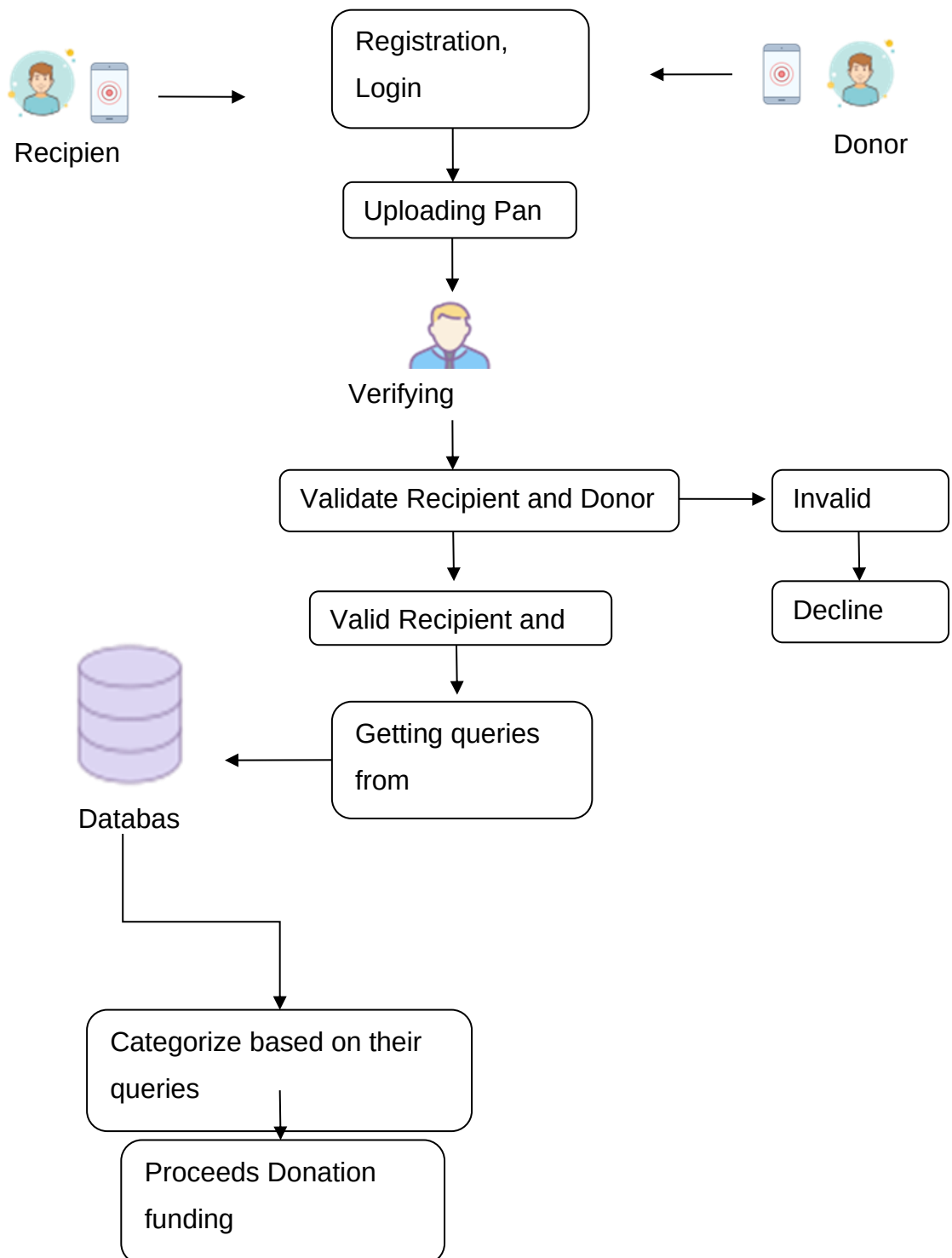


Fig: 5.1 SYSTEM DESIGN

5.2 SEQUENCE DIAGRAM:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

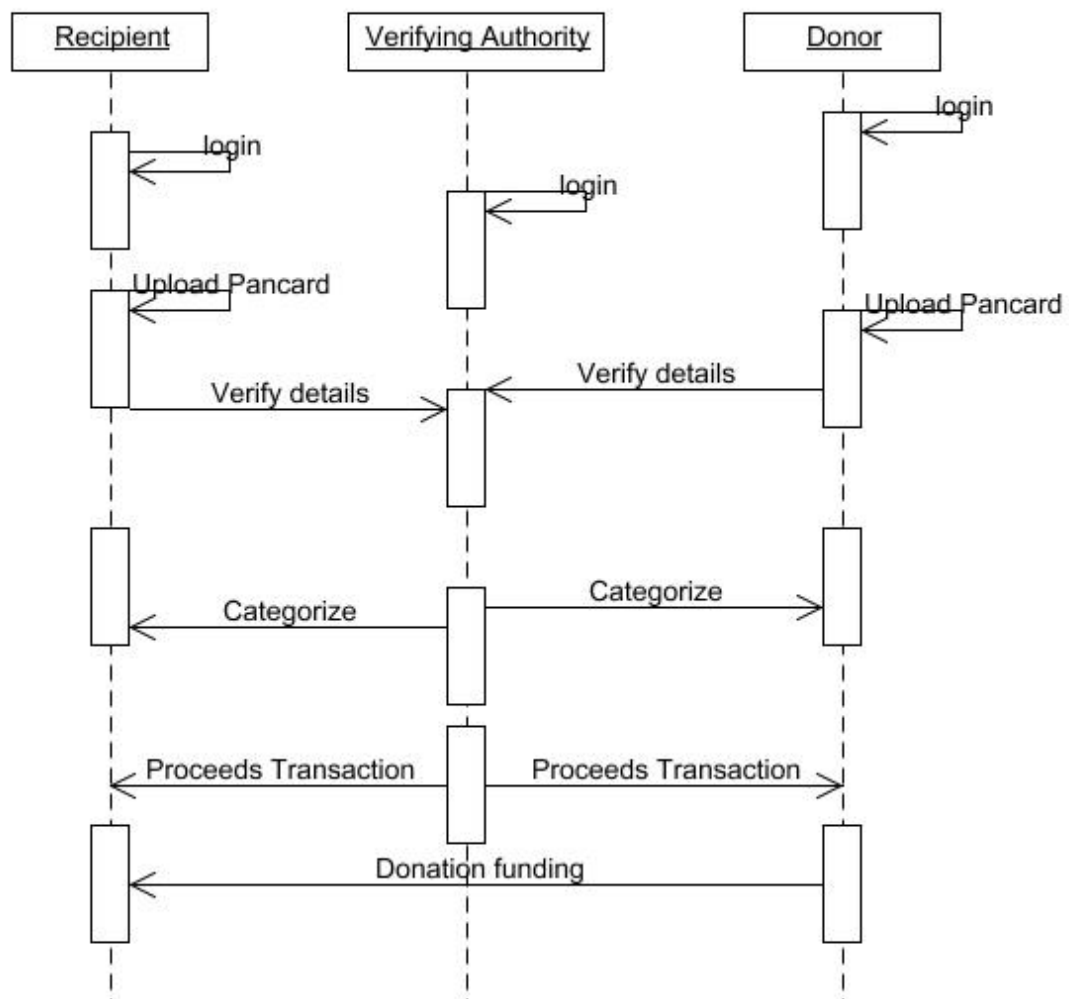


Fig.5.2 SEQUENCE DIAGRAM

5.3 USE CASE DIAGRAM:

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

5.3.1. USECASE DIAGRAM:

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.

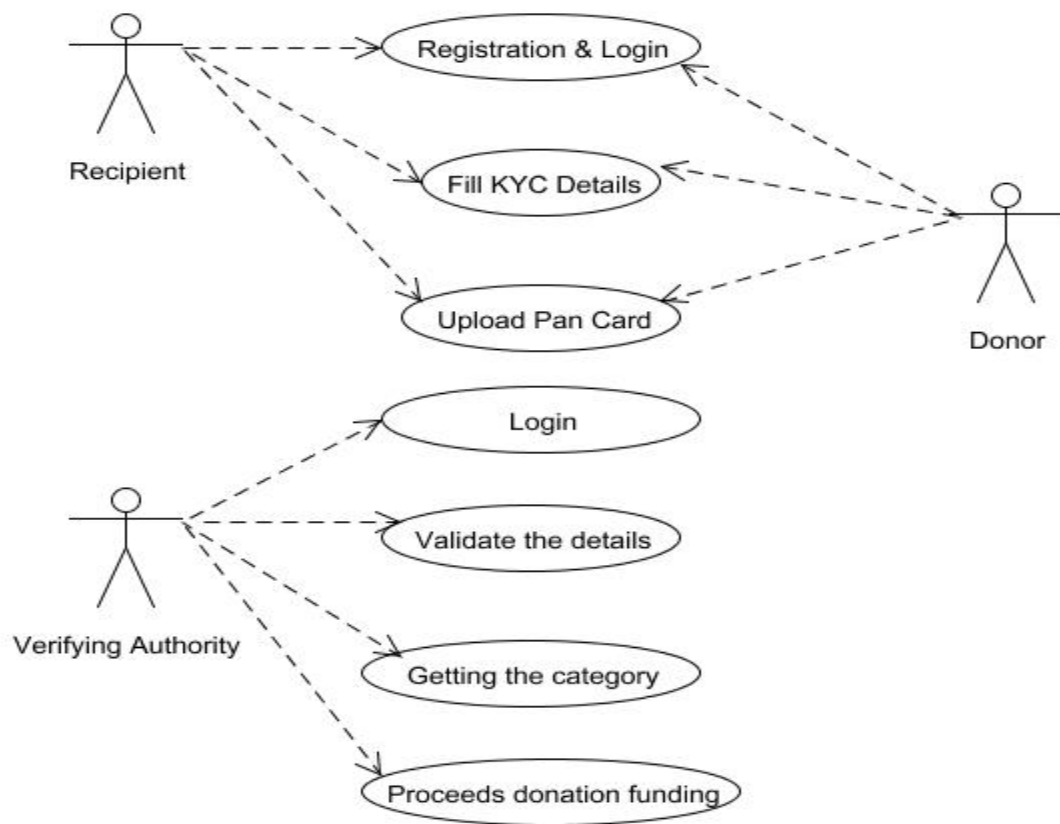


Fig.5.3 USE CASE DIAGRAM

5.4 ACTIVITY DIAGRAM:

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.

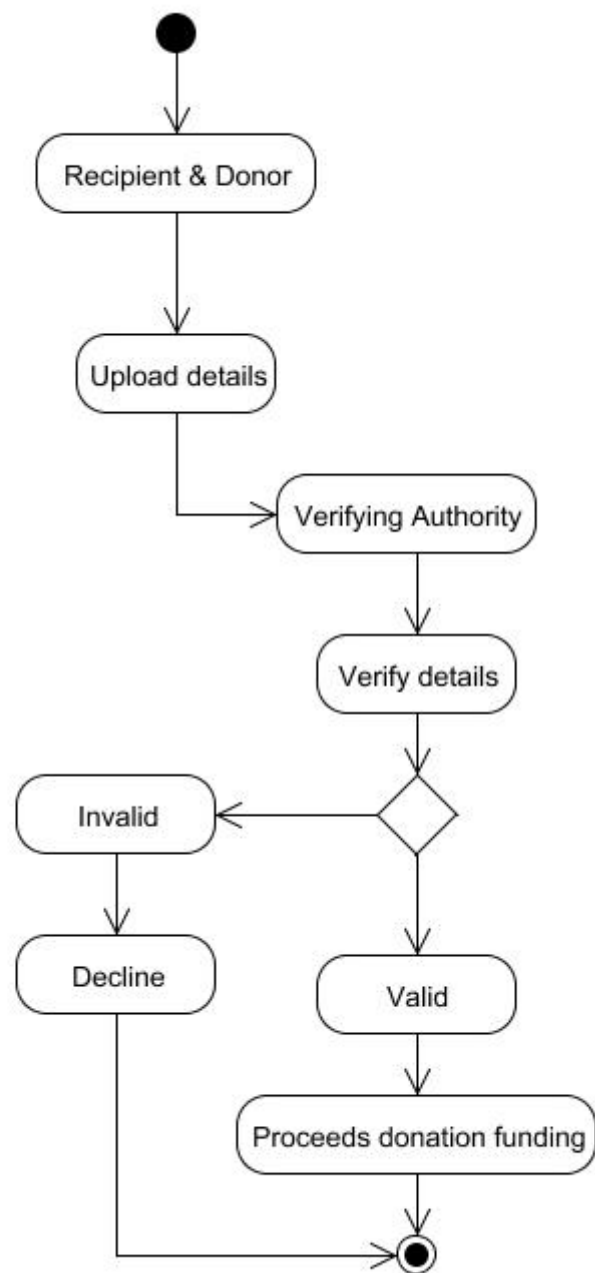


Fig.5.4 ACTIVITY DIAGRAM

5.5 COLLABORATION DIAGRAM:

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist.

The collaboration diagram illustrates messages being sent between classes and objects.

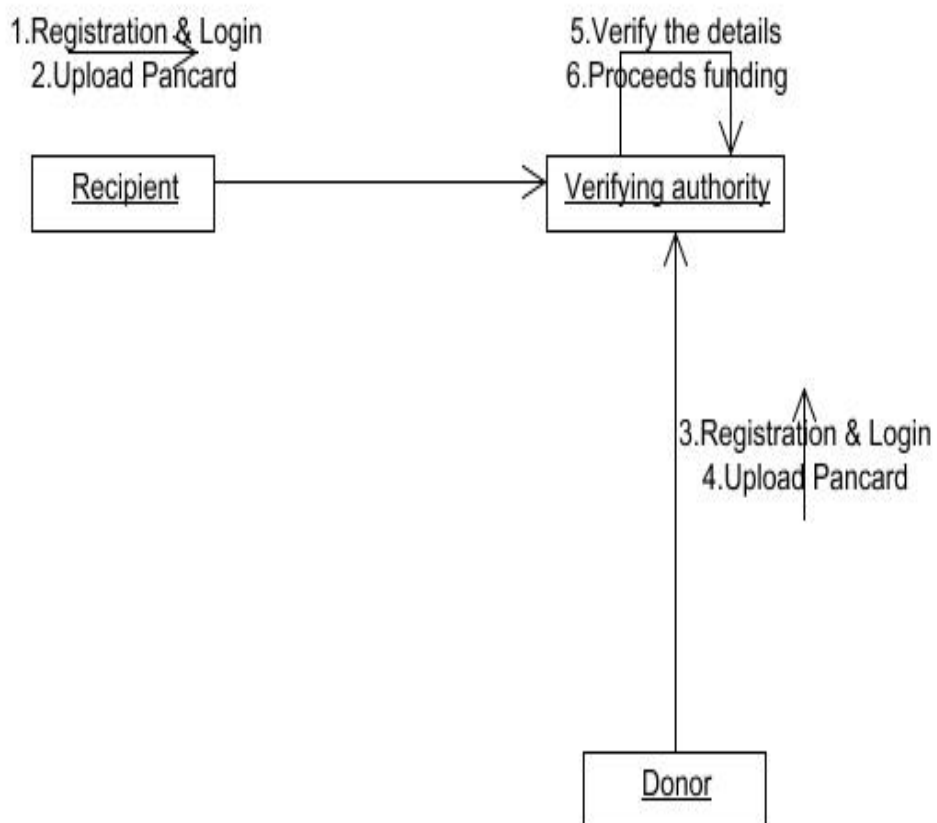
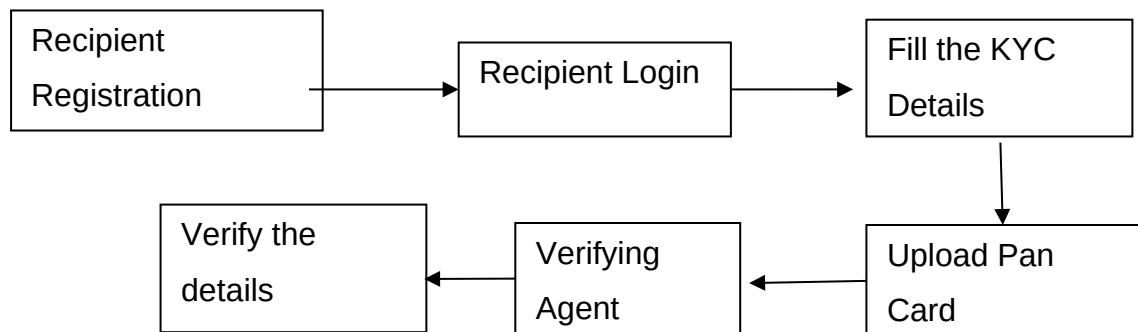


Fig.5.5 COLLABORATION DIAGRAM

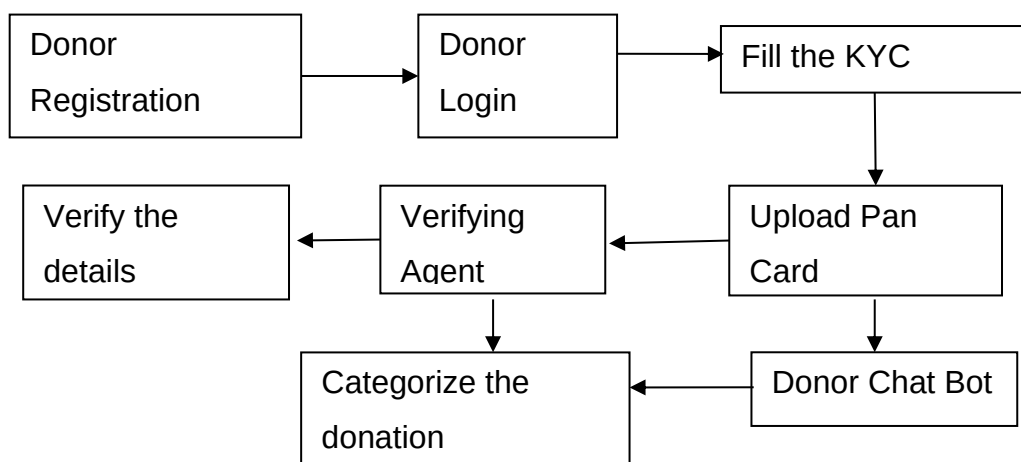
5.6 DATA FLOW DIAGRAM:

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modelling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

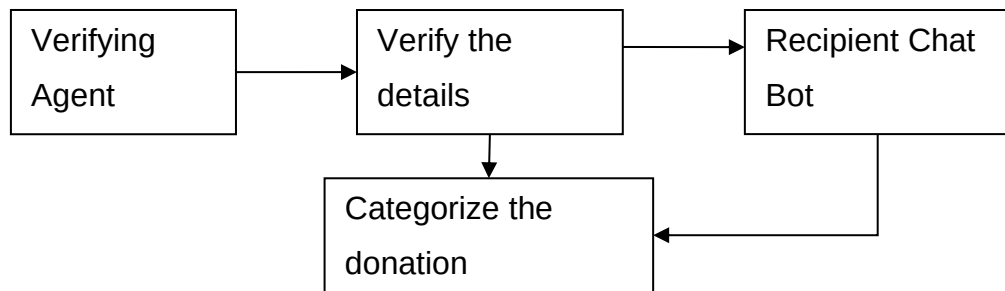
Level 0:



Level 1:



Level 2:



Level 3:

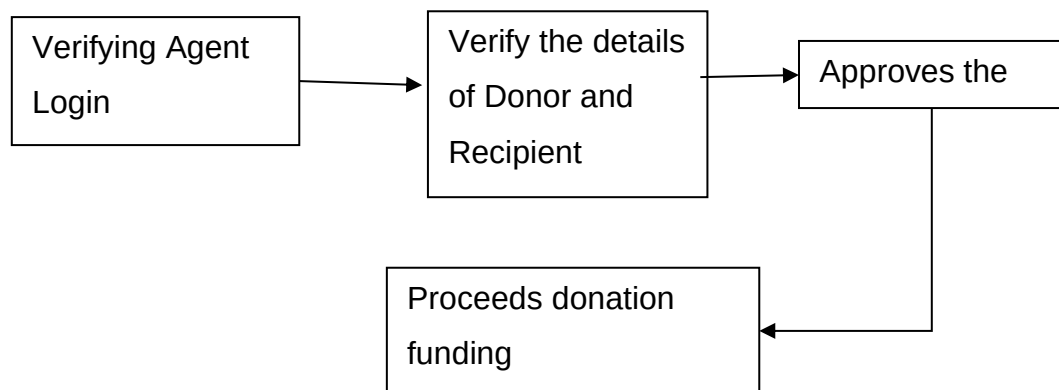


Fig.5.6 DATA FLOW DIAGRAM

5.7 CLASS DIAGRAM:

A Class diagram in the Unified Modelling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

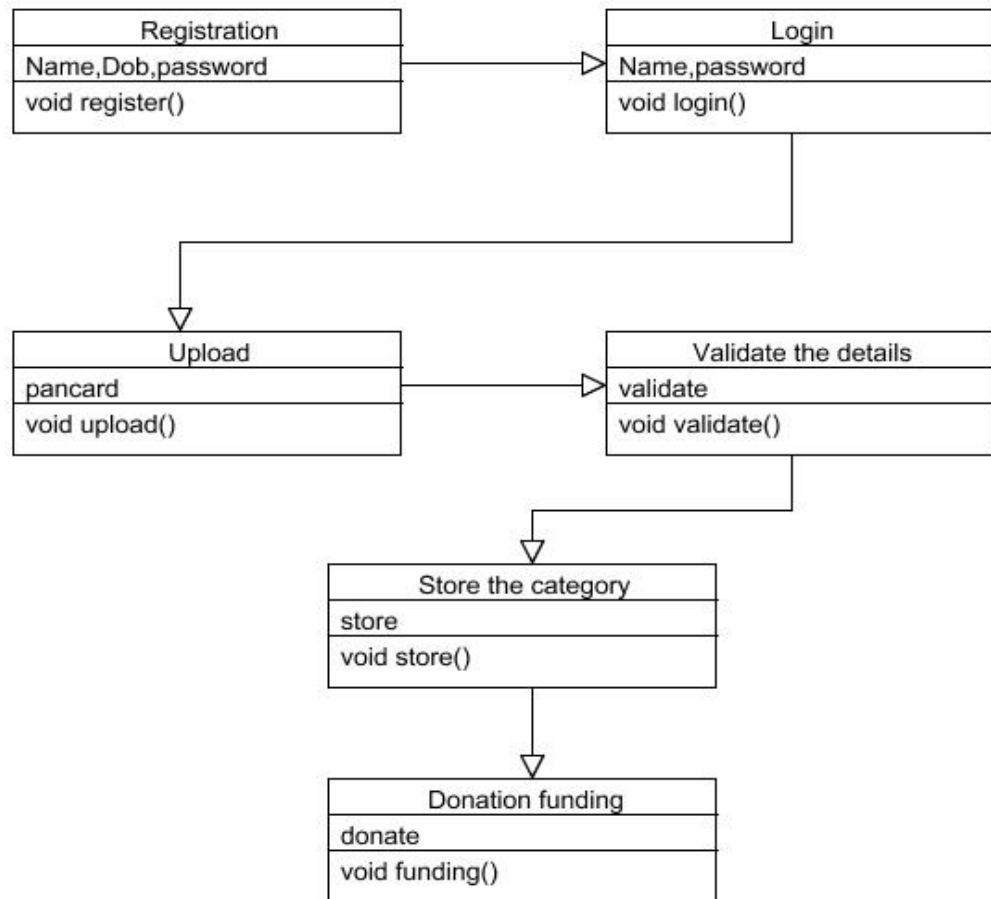


Fig.5.7 CLASS DIAGRAM

CHAPTER 6

SYSTEM DESIGN

The recipient has to fill the registration form with required mandatory input fields and the personal details completely. These details will get further verified by the government authority or the verifying agent appointed by the government. The donor has to fill the registration form with required mandatory input fields and the personal details completely. These details will get further verified by the government authority or the verifying agent appointed by the government. By using the token,

generated after the completion of verifying recipient's details by the government authority. The recipient will login only using the unique token to request money. Their request contains the details of the category for which the recipient needs money. The recipient can view their request details and status of donation for their requests. By using the token, generated after the completion of verifying donor's details by the government authority. The donor will login only using the unique token to donate money. Their request contains the details or the category for which the donor is willing to donate. The donor can view the history of their donations. After verifying all the details of donor and the recipient, the verifying person will approve the details of recipient and the donor and further allows the process of funding. After the funding process, the donor will be provided a certificate by the government. The donor will get notifications automatically in order to appreciate their donation funding.

6.1 MODULES

- Donor Verification
- Donor Verification and Donor's Chatbot
- Recipient's Chatbot
- Providing fund from donor to recipient

6.2 MODULE EXPLANATION:

6.2.1 DONOR VERIFICATION:

The donor has to fill the registration form with required mandatory input fields and the personal details completely. These details will get verified by the government authority or the verifying agent appointed by the government. After successful verification, the donor will get the registration and the authenticated mail.

RECIPIENT VERIFICATION AND CHATBOT:

The recipient has to fill the registration form with required mandatory input fields and the personal details completely. These details will get verified by the government authority or the verifying agent appointed by the government as same as the donor's verification. After successful verification, the recipient will get the

notifications regarding their donation with secured authentication. The recipient's chatbot contains the details or the category for which the recipient needs donation.

DONOR'S CHATBOT:

The donor's chatbot contains the details or the category for which the donor is willing to donate money. The details of the recipients for which the donor request to donate will get displayed.

PROVIDING FUND FROM DONOR TO RECIPIENT:

After verifying all the details of donor and the recipient, the verifying person will approve the details of recipient and the donor and further allows the process of funding. The donor will be displayed with the recipient details in the category for which the donor is willing to donate. The donor will select the recipient and further donates. After the funding process, the donor will be provided by a certificate by the government.

CHAPTER 7

CODING AND TESTING

7.1 CODING:

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

7.2 CODING STANDARDS:

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in

such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

Exception and error handling

7.2.1 NAMING CONVENTIONS:

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

7.2.2 VALUE CONVENTIONS:

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.

- Proper documentation of flag values.

7.2.3 SCRIPT WRITING AND COMMENTING STANDARD:

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

7.2.4 MESSAGE BOX FORMAT:

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

7.3 TEST PROCEDURE:

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

7.4 TEST DATA AND OUTPUT:

7.4.1 UNIT TESTING:

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

7.4.2 FUNCTIONAL TESTS:

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

7.4.3 PERFORMANCE TEST:

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

7.4.4 STRESS TEST:

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.4.5 STRUCTURED TEST:

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

7.4.6 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface discrepancies can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.5 TESTING TECHNIQUES / TESTING STRATEGIES:

7.5.1 TESTING:

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

7.5.1.1 WHITE BOX TESTING:

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases

- Graph matrices Control

7.5.1.2 BLACK BOX TESTING:

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

7.5.2 SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

7.5.2.1 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors,

should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

7.5.2.2 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

7.5.2.3 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

7.5.2.4 VALIDATION TESTING:

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer.

Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- * The function or performance characteristics confirm to specifications and are accepted.
- * A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

7.5.2.5 USER ACCEPTANCE TESTING:

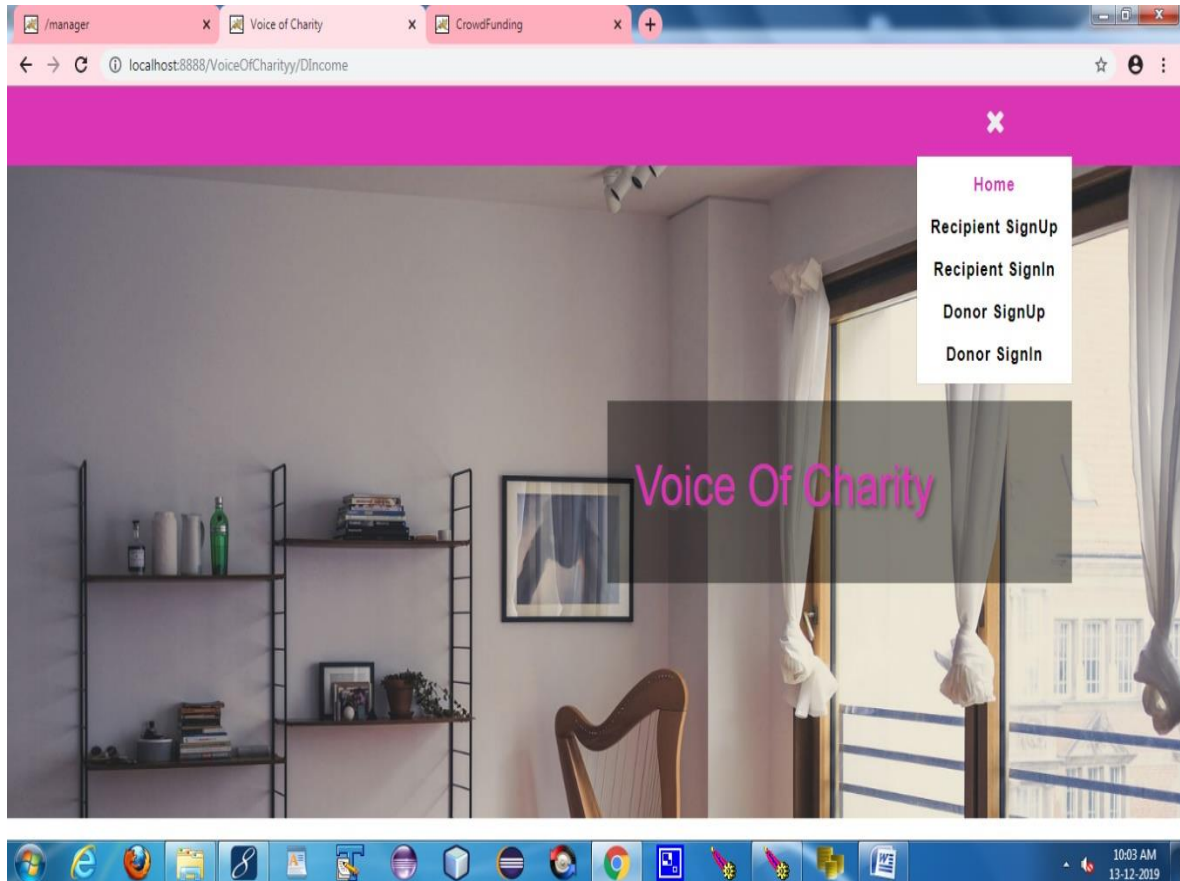
User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

CHAPTER-8

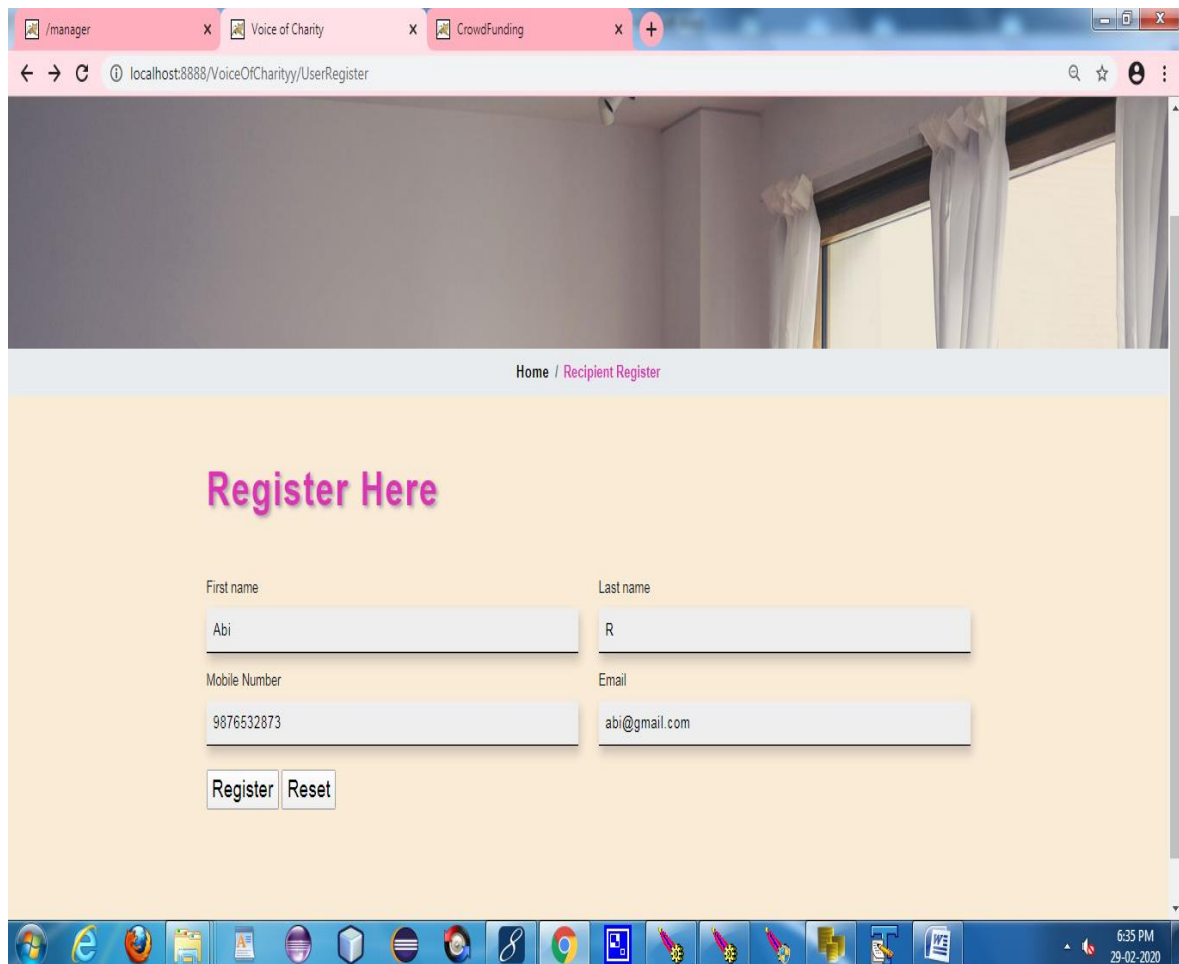
RESULT AND DISCUSSION

8.1 HOME PAGE



The **Fig.8.1** is the home page the donor has to fill the registration form with required mandatory input fields and the personal details completely. These details will get verified by the government authority or the verifying agent appointed by the government. After successful verification, the donor will get the registration and the authenticated mail.

8.2 RECIPIENT REGISTER



The screenshot displays a web browser window with three tabs: '/manager', 'Voice of Charity', and 'CrowdFunding'. The address bar shows 'localhost:8888/VoiceOfCharity/UserRegister'. The page features a header with a navigation bar containing 'Home' and 'Recipient Register'. Below the header is a large orange section with the text 'Register Here' in pink. The registration form contains four input fields: 'First name' (Abi), 'Last name' (R), 'Mobile Number' (9876532873), and 'Email' (abi@gmail.com). At the bottom of the form are two buttons: 'Register' and 'Reset'.

The **Fig.8.2** recipient has to fill the registration form with required mandatory input fields and the personal details completely. These details will get verified by the government authority or the verifying agent appointed by the government as same as the donor's verification. After successful verification, the recipient will get the notifications regarding their donation with secured authentication. The recipient's chatbot contains the details or the category for which the recipient needs donation.

8.3 PROVIDING FUND FROM DONOR TO RECIPIENT

ID	Name	Image	Email	Expected date for money need	Required amount of money	Pay
UID_825	Vivekananda		Vivekananda@gmail.com	2020-02-19	30000 to go	Lend Now
UID_723	Tarun		Tarun1@gmail.com	null	null to go	Lend Now
UID_132	Arindam		Arindam1@gmail.com	null	null to go	Lend Now
UID_1918	Akshit		Akshit@gmail.com	2020-11-25	30000 to go	Lend Now
Female Recipients						
UID_850	Debjyoti		Debjyoti1@gmail.com	2020-04-22	20000 to go	1000 Lend Now
UID_73	Kaustav		Kaustav@gmail.com	2020-05-27	20000 to go	Lend Now

The **Fig.8.3** After verifying all the details of donor and the recipient, the verifying person will approve the details of recipient and the donor and further allows the process of funding. The donor will be displayed with the recipient details in the category for which the donor is willing to donate. The donor will select the recipient and further donates. After the funding process, the donor will be provided by a certificate by the government.

CHAPTER-9

CONCLUSION AND FUTURE WORK

CONCLUSION

In this paper, we presented a focused study on prospecting the donation careers in crowdfunding. By collecting and analysing large-scale real-world data, we specifically formalized predicting tasks by automatically generating notifications to

the donors, in order to appreciate their process of donation funding. Then, using a clustering algorithm and K-Nearest neighbour algorithm, we proposed a model which could integrate heterogeneous features to jointly model the donor activities in predicting their donation interests. Additionally, we designed multiple innovative constraints and incorporated them into objective functions for modelling the censoring phenomenon and dependence relations of different behaviours. In experiments, we analysed the donations in crowdfunding and validated the prediction performances from various aspects. The experimental results clearly demonstrated the effectiveness of our proposed models for analysing and predicting the donors and appreciating them for further more donations with a trustworthy funding.

FUTURE WORK

Our study may bring some new insights from the application view of crowdfunding and the technical view of exploiting deep learning for survival analysis to the research communities. In the future, we will apply and improve our models for other scenarios, such as traditional charity activities, especially applied to survival data with modelling collaborative tasks in some other domains, such as device failure modelling in engineering, predicting student dropout, and prospecting the career development.

REFERENCES:

- [1] H. Zhao, Y. Ge, Q. Liu, G. Wang, E. Chen, and H. Zhang, "P2p lending survey: Platforms, recent advances and prospects," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 6, p. 72, 2017.
- [2] H. Zhao, H. Zhang, Y. Ge, Q. Liu, E. Chen, H. Li, and L. Wu, "Tracking the dynamics in crowdfunding," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 625–634.
- [3] J. Choo, D. Lee, B. Dilkina, H. Zha, and H. Park, "To gather together for a better world: Understanding and leveraging communities in micro-lending recommendation," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 249–260.

- [4] T. Althoff and J. Leskovec, "Donor retention in online crowdfunding communities: A case study of donorschoose. org," in Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2015, pp. 34–44.
- [5] L. J. Rosenberg and J. A. Czepiel, "A marketing approach for on Knowledge and Data Engineering, vol. 28, no. 4, pp. 901–911, 2016. [12] G. Li, Y. Zheng, J. Fan, J. Wang, and R. Cheng, "Crowdsourced data management: Overview and challenges," in Proceedings of the 2017 ACM International Conference on Management of Data. ACM, 2017, pp. 1711–1716.
- [13] A. I. Chittilappilly, L. Chen, and S. Amer-Yahia, "A survey of general-purpose crowdsourcing techniques," IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 9, pp. 2246–2266, 2016.
- [14] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 9, pp. 2296–2319, 2016.
- [15] G. Li, J. Wang, Y. Zheng, J. Fan, and M. J. Franklin, Crowdsourced Data Management: Hybrid Human-Machine Data Management. Springer, 2018.
- [16] Y. Zheng, R. Cheng, S. Maniu, and L. Mo, "On optimality of jury selection in crowdsourcing," in Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015. Open-Proceedings. org., 2015.
- [17] Y. Zheng, G. Li, and R. Cheng, "Docs: a domain-aware crowdsourcing system using knowledge bases," Proceedings of the VLDB Endowment, vol. 10, no. 4, pp. 361–372, 2016.
- [18] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng, "Truth inference in crowdsourcing: Is the problem solved?" Proceedings of the VLDB Endowment, vol. 10, no. 5, pp. 541–552, 2017.
- [19] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng, "Crowdsourced poi labelling: Location-aware result inference and task assignment," in Data Engineering (ICDE), 2016 IEEE 32nd International Conference on. IEEE, 2016, pp. 61–72.
- [20] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "Qasca: A qualityaware task assignment system for crowdsourcing applications," in Proceedings of the 2015 ACM SIGMOD international conference on management of data. ACM, 2015, pp. 1031–1046.

- [21] C.-T. Lu, S. Xie, X. Kong, and P. S. Yu, "Inferring the impacts of social media on crowdfunding," in Proceedings of the 7th ACM international conference on Web search and data mining. ACM, 2014, pp. 573–582.
- [22] H. Zhang, H. Zhao, Q. Liu, T. Xu, E. Chen, and X. Huang, "Finding potential lenders in p2p lending: A hybrid random walk approach," Information Sciences, vol. 432, pp. 376–391, 2018.
- [23] H. Zhao, Q. Liu, G. Wang, Y. Ge, and E. Chen, "Portfolio selections in p2p lending: A multi-objective perspective," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016, pp. 2075–2084.
- [24] V. Rakesh, W.-C. Lee, and C. K. Reddy, "Probabilistic group recommendation model for crowdfunding domains," in Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 2016, pp. 257–266.
- [25] Y. Li, V. Rakesh, and C. K. Reddy, "Project success prediction in crowdfunding environments," in Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 2016, pp. 247–256.
- [26] H. Zhao, Q. Liu, H. Zhu, Y. Ge, E. Chen, Y. Zhu, and J. Du, "A sequential approach to market state modeling and analysis in online p2p lending," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 1, pp. 21–33, 2018.
- [27] J. An, D. Quercia, and J. Crowcroft, "Recommending investors for crowdfunding projects," in Proceedings of the 23rd international conference on World wide web. ACM, 2014, pp. 261–270.
- [28] Q. Liu, G. Wang, H. Zhao, C. Liu, T. Xu, and E. Chen, "Enhancing campaign design in crowdfunding: A product supply optimization perspective," in Proceedings of the 26th International Joint Conference on Artificial Intelligence, ser. IJCAI'17. AAAI Press, 2017, pp. 695–702.
- [29] J. Choo, C. Lee, D. Lee, H. Zha, and H. Park, "Understanding and promoting micro-finance activities in kiva. org," in Proceedings of the 7th ACM international conference on Web search and data mining. ACM, 2014, pp. 583–592.
- [30] T. Iwata, K. Saito, and T. Yamada, "Recommendation method for improving customer lifetime value," IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 9, pp. 1254–1263, 2008.

- [31] H. Jing and A. J. Smola, “Neural survival recommender,” in Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. ACM, 2017, pp. 515–524.
- [32] D. R. Cox, “Regression models and life-tables,” in Breakthroughs in statistics. Springer, 1992, pp. 527–541.

SOURCE CODE

```
package com.banklogic;

import java.io.ByteArrayInputStream;
import java.io.InputStream;

import com.opensymphony.xwork2.ActionSupport;

public class AjaxSource extends ActionSupport implements DataMapInterface
{
    public InputStream inputstream;
    String hidden,key;

    public String getKey() {
        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }

    public String getHidden() {
        return hidden;
    }

    public void setHidden(String hidden) {
        this.hidden = hidden;
    }

    public InputStream getInputStream()
    {
        return inputstream;
    }

    public void setInputStream(InputStream inputstream)
    {
        this.inputstream = inputstream;
    }

    public String execute()
    {
        try
        {
            if(hidden.equals("cityname"))
            {
                if(key==null || key.equals("")) ||
!pincodelistmap.containsKey(key))
                {

```

```

                                inputstream=new
ByteArrayInputStream(("nil&nil&pincodeajax").trim().getBytes());
                                }
                                else
                                {
                                    String spl[]=pinodelistmap.get(key).split("\\*");
                                    inputstream=new
ByteArrayInputStream((spl[0]+"&"+spl[2]+"&pincodeajax").trim().getBytes());

                                }
                            }
                            else
                            if(hidden.equals("editholder"))
                            {
                                DbQueries db=new DbQueries();
                                String str=db.getUserAccountInd(key);
                                inputstream=new
ByteArrayInputStream((str+"&acctinfoajax").trim().getBytes());
                            }
                            else
                            if(hidden.equals("addamountholderaccount"))
                            {
                                DbQueries db=new DbQueries();
                                String str=db.retAccountAmount(key);
                                inputstream=new
ByteArrayInputStream((str+"&addamountinfoajax").trim().getBytes());
                            }
                        }
                        catch(Exception e)
                        {
                            e.printStackTrace();
                        }
                        return "success";
                    }
                }
            }
}

```

```
package com.banklogic;
```

```
import java.io.ByteArrayInputStream;
```

```
import java.io.InputStream;
```

```
import com.opensymphony.xwork2.ActionSupport;
```

```
public class AjaxSource extends ActionSupport implements DataMapInterface
{
```

```
    public InputStream inputstream;
```

```
    String hidden,key;
```

```
    public String getKey() {
```

```

        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }

    public String getHidden() {
        return hidden;
    }

    public void setHidden(String hidden) {
        this.hidden = hidden;
    }

    public InputStream getInputStream()
    {
        return inputstream;
    }

    public void setInputStream(InputStream inputstream)
    {
        this.inputstream = inputstream;
    }

    public String execute()
    {
        try
        {
            if(hidden.equals("cityname"))
            {
                if(key==null || key.equals("")) ||
!pincodelistmap.containsKey(key))
                {
                    inputstream=new
ByteArrayInputStream(("nil&nil&pincodeajax").trim().getBytes());
                }
                else
                {
                    String spl[]=pincodelistmap.get(key).split("\\*");
                    inputstream=new
ByteArrayInputStream(("spl[0]+"&"+spl[2]+"&pincodeajax").trim().getBytes());
                }
            }
            else
            if(hidden.equals("editholder"))
            {
                DbQueries db=new DbQueries();
                String str=db.getUserAccountInd(key);
            }
        }
    }

```

```

        inputstream=new
ByteArrayInputStream((str+"&acctinfoajax").trim().getBytes());
    }
    else
    if(hidden.equals("addamountholderaccount"))
    {
        DbQueries db=new DbQueries();
        String str=db.retAccountAmount(key);
        inputstream=new
ByteArrayInputStream((str+"&addamountinfoajax").trim().getBytes());
    }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return "success";
}
}

```

```

package com.banklogic;

```

```

import java.io.Serializable;

```

```

public class CommonBean implements Serializable

```

```

{
    String accountnumb,username,mobnum,email,address,city,pincode,state;
    String amount,pinpass,opass;
    String sessionId,merchantname,tknId,productid="";

    public String getTknId() {
        return tknId;
    }

    public void setTknId(String tknId) {
        this.tknId = tknId;
    }

    public String getSessionId() {
        return sessionId;
    }

    public void setSessionId(String sessionId) {
        this.sessionId = sessionId;
    }

    public String getMerchantname() {

```

```

        return merchantname;
    }

    public void setMerchantname(String merchantname) {
        this.merchantname = merchantname;
    }

    public String getProductid() {
        return productid;
    }

    public void setProductid(String productid) {
        this.productid = productid;
    }

    public String getOpass() {
        return opass;
    }

    public void setOpass(String opass) {
        this.opass = opass;
    }

    public String getPinpass() {
        return pinpass;
    }

    public void setPinpass(String pinpass) {
        this.pinpass = pinpass;
    }

    public String getAmount() {
        return amount;
    }

    public void setAmount(String amount) {
        this.amount = amount;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getAccountnumb() {
        return accountnumb;
    }

```

```
public void setAccountnumb(String accountnumb) {
    this.accountnumb = accountnumb;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getMobnum() {
    return mobnum;
}

public void setMobnum(String mobnum) {
    this.mobnum = mobnum;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public String getCity() {
    return city;
}

public void setCity(String city) {
    this.city = city;
}

public String getPincode() {
    return pincode;
}

public void setPincode(String pincode) {
    this.pincode = pincode;
}

public String getState() {
    return state;
}

public void setState(String state) {
```

```

        this.state = state;
    }
}

package com.banklogic;

import java.util.LinkedHashMap;

public interface DataMapInterface
{
    LinkedHashMap<String,String> pincodelistmap=new
    LinkedHashMap<String,String>();
}

package com.banklogic;

import java.sql.Connection;
import java.sql.DriverManager;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class DbListener implements ServletContextListener
{
    Connection conn=null;
    public void contextDestroyed(ServletContextEvent arg0)
    {
        try
        {
            conn.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    public void contextInitialized(ServletContextEvent arg1)
    {
        ServletContext sc=arg1.getServletContext();
        try
        {
            Class.forName("com.mysql.jdbc.Driver").newInstance();

            conn=DriverManager.getConnection("jdbc:mysql://localhost/bankadmindb","
root","root");

            sc.setAttribute("connection",conn);
            System.out.println("BankAdmin Connection Created");
        }
    }
}

```

```

        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

```
package com.banklogic;
```

```

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

```

```

import javax.servlet.ServletContext;
import org.apache.struts2.ServletActionContext;
import com.mysql.jdbc.exceptions.MySQLIntegrityConstraintViolationException;

```

```

public class DbQueries
{

```

```

    String qu="";
    Connection con=null;
    PreparedStatement pst=null;

```

```

    public DbQueries()
    {
        ServletContext sc=ServletActionContext.getServletContext();
        con=(Connection) sc.getAttribute("connection");
    }

```

```

    public Integer insertUserDetails(CommonBean comb)
    {
        int dbst=0;
        try
        {
            pst=con.prepareStatement("insert into userinfo
values(?,?,?,?,?,?,?,?,?)");
            pst.setString(1,comb.getAccountnumb());
            pst.setString(2,comb.getUsername());
            pst.setString(3,comb.getMobnum());
            pst.setString(4,comb.getEmail());
            pst.setString(5,comb.getAddress());
            pst.setString(6,comb.getCity());
            pst.setString(7,comb.getPincode());
            pst.setString(8,comb.getState());
            dbst=pst.executeUpdate();

```

```

        pst=con.prepareStatement("insert into useraccountinfo
values(?,?,?,?,?)");
        pst.setString(1,comb.getAccountnumb());
        pst.setString(2,comb.getPinpass());
        pst.setString(3,comb.getAmount());
        pst.setString(4,"null");
        pst.setString(5,"null");
        dbst=pst.executeUpdate();
        pst.close();
    }
    catch(MySQLIntegrityConstraintViolationException e)
    {
        dbst=0;
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    return dbst;
}

public ArrayList getUsersAccountld()
{
    ArrayList arrayl=new ArrayList();
    try
    {
        qu="select accountno from userinfo";
        pst=con.prepareStatement(qu);
        ResultSet rs=pst.executeQuery();
        while(rs.next())
        {
            arrayl.add(rs.getString(1));
        }
        pst.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    return arrayl;
}

public String getUserAccountlnd(String accno)
{
    String str="";
    try
    {
        qu="SELECT userinfo.*,useraccountinfo.amount"
        +" FROM userinfo"

```

```

        +" INNER JOIN useraccountinfo"
        +" ON userinfo.accountno=useraccountinfo.accountno"
        +" where userinfo.accountno=?";

        pst=con.prepareStatement(qu);
        pst.setString(1,accno);
        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {

            str=accno+"&"+rs.getString(2)+"&"+rs.getString(9)+"&"+rs.getString(3)+"&"+
            rs.getString(4)+"&"+rs.getString(5)+"&"+rs.getString(6)+"&"+rs.getString(7)+"&"+rs
            .getString(8);
        }
        pst.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    return str;
}

public String retAccountAmount(String accno)
{
    String str="";
    try
    {
        pst=con.prepareStatement("select accountholdername from
        userinfo where accountno=?");
        pst.setString(1,accno);
        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {
            str=accno+"&"+rs.getString(1);
        }
        pst.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    return str;
}

public void updateAccountHolderTab(CommonBean comb)
{
    try
    {

```

```

        pst=con.prepareStatement("update userinfo set
accountholdername=?,mobilenumber=?,mailid=?,Address=?,city=?,pincode=?,stat
e=? where accountno=? ");
        pst.setString(1,comb.getUsername());
        pst.setString(2,comb.getMobnum());
        pst.setString(3,comb.getEmail());
        pst.setString(4,comb.getAddress());
        pst.setString(5,comb.getCity());
        pst.setString(6,comb.getPincode());
        pst.setString(7,comb.getState());
        pst.setString(8,comb.getAccountnumb());
        pst.executeUpdate();
        pst.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

public void addAmountAccount(CommonBean comb)
{
    try
    {
        String str="";
        pst=con.prepareStatement("select amount from
useraccountinfo where accountno=?");
        pst.setString(1,comb.getAccountnumb());
        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {
            str=rs.getString(1);
        }

        int
am=Integer.parseInt(str.trim())+Integer.parseInt(comb.getAmount().toString().trim()
);

        pst=con.prepareStatement("update useraccountinfo set
amount=? where accountno=?");
        pst.setString(1,(am+""));
        pst.setString(2,comb.getAccountnumb());
        pst.executeUpdate();
        pst.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

```

```

public String checkTransaction(CommonBean comb)
{
    int dbst=0;
    String stats="";
    String transamt="";
    try
    {
        pst=con.prepareStatement("select
pinpass,amount,opass,transferamount from useraccountinfo where
accountno=?");
        pst.setString(1,comb.getAccountnumb());
        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {
            String atmpin=rs.getString(1);
            String amount=rs.getString(2);
            String opass=rs.getString(3);
            transamt=rs.getString(4);

            if(atmpin.equals(comb.getPinpass()) &&
opass.equals(comb.getOpass()))
            {
                int amt=Integer.parseInt(amount)-
Integer.parseInt(transamt);
                if(amt>0)
                {
                    String
status=amuntupdate(comb.getMerchantname(),transamt);
                    if(status.equalsIgnoreCase("Success"))
                    {
                        String
stat=transactionupdate(comb.getAccountnumb(),comb.getMerchantname(),comb.
getProductid(),transamt,status,comb.getTknId());

                        if(stat.equalsIgnoreCase("success"))
                        {

                            pst=con.prepareStatement("update useraccountinfo set amount=? where
accountno=?");
                            pst.setString(1,""+amt);

                            pst.setString(2,comb.getAccountnumb());
                            dbst=pst.executeUpdate();

                            pst=con.prepareStatement("update useraccountinfo set transferamount=?
where accountno=?");
                            pst.setString(1,"null");

```

```

        pst.setString(2,comb.getAccountnumb());
                                pst.executeUpdate();
                                }
                                else{
                                    System.out.println();

        transactionupdate(comb.getAccountnumb(),comb.getMerchantname(),com
b.getProductid(),amount,"failure",comb.getTknld());
                                stats="failure";
                                }
                                }
                                else
                                {

        transactionupdate(comb.getAccountnumb(),comb.getMerchantname(),com
b.getProductid(),amount,"failure",comb.getTknld());
                                stats="failure";
                                }
                                }
                                else
                                {
                                    System.out.println("Low Balance....");
                                    stats="low";
                                }
                                }
                                else{
                                    stats="Wpin";
                                    System.out.println("OTP Or Pin Wrong...");
                                }
                                }
                                }
                                catch (Exception e)
                                {
                                    e.printStackTrace();
                                }
                                if(dbst!=0)
                                {
                                    stats="success";
                                }
                                return stats+"&"+transamt;
                                }
                                public String amuntupdate(String accholdnam,String transamt)
                                {
                                    String status="",accno="";
                                    int totAmt=0;
                                    try
                                    {
                                        pst=con.prepareStatement("select accountno from userinfo
where accountholdername=?");

```

```

        pst.setString(1,accholdnam);
        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {
            accno=rs.getString(1).toString().trim();
        }
        pst=con.prepareStatement("select amount from
useraccountinfo where accountno=?");
        pst.setString(1,accno);
        ResultSet rs1=pst.executeQuery();
        if(rs1.next())
        {
            String amt=rs1.getString(1).toString().trim();

            totAmt=Integer.parseInt(amt)+Integer.parseInt(transamt);
        }
        pst=con.prepareStatement("update useraccountinfo set
amount=? where accountno=?");
        pst.setString(1,String.valueOf(totAmt));
        pst.setString(2,accno);
        pst.executeUpdate();
        status="success";

    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return status;
}

public String transactionupdate(String accnum,String accname,String
prodID,String proAmt,String sts,String tkID)
{
    try
    {
        SimpleDateFormat sdf=new SimpleDateFormat("yyy-MM-dd
HH:mm:ss");
        Date now=new Date();
        String strdate=sdf.format(now);
        System.out.println(strdate);
        pst=con.prepareStatement("insert into tokengeneration
values(?,?,?,?,?,?,?)");
        pst.setString(1,accnum);
        pst.setString(2,accname);
        pst.setString(3,strdate);
        pst.setString(4,prodID);
        pst.setString(5,proAmt);
        pst.setString(6,tkID);
        pst.setString(7,sts);
        pst.executeUpdate();
    }
}

```

```

    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return "success";
}
}

```

```
package com.banklogic;
```

```
import java.io.FileInputStream;
import java.sql.ResultSet;
import java.util.Properties;
import java.util.StringTokenizer;
```

```
import com.mysql.jdbc.PreparedStatement;
import com.opensymphony.xwork2.ActionSupport;
```

```
public class UserPaymentProc extends ActionSupport
{
    String firstbankholdernames,firstamtpin,firstopass;
    String url,payhidden;
    String sessionId,merchantname,productid="";

    public String getMerchantname() {
        return merchantname;
    }

    public void setMerchantname(String merchantname) {
        this.merchantname = merchantname;
    }

    public String getProductid() {
        return productid;
    }

    public void setProductid(String productid) {
        this.productid = productid;
    }

    public String getSessionId() {
        return sessionId;
    }

    public void setSessionId(String sessionId) {
        this.sessionId = sessionId;
    }
}

```

```

public String getPayhidden() {
    return payhidden;
}

public void setPayhidden(String payhidden) {
    this.payhidden = payhidden;
}

public String getUrl() {
    return url;
}

public void setUrl(String url) {
    this.url = url;
}

public String getFirstbankholdernames() {
    return firstbankholdernames;
}

public void setFirstbankholdernames(String firstbankholdernames) {
    this.firstbankholdernames = firstbankholdernames;
}

public String getFirstamtpin() {
    return firstamtpin;
}

public void setFirstamtpin(String firstamtpin) {
    this.firstamtpin = firstamtpin;
}

public String getFirstopass() {
    return firstopass;
}

public void setFirstopass(String firstopass) {
    this.firstopass = firstopass;
}

public String execute()
{
    String status="redirect";
    try
    {

        System.out.println("=====*=====
"+sessionId);

        DbQueries db=new DbQueries();
        long tkn=(long) Math.floor((Math.random()*89999)+10000);

```

```

String tkID="Tk"+tkn;
CommonBean com=new CommonBean();
com.setAccountnumb(firstbankholdernames);
com.setPinpass(firstamtpin);
com.setOpass(firstopass);
com.setSessionId(sessionId);
com.setMerchantname("webserver"+merchantname);
com.setProductid(productid);
com.setTknId(tkID);
String dbstamt=db.checkTransaction(com);
System.out.println(dbstamt);
StringTokenizer stt=new StringTokenizer(dbstamt, "&");
String dbst=stt.nextToken().toString();
String amt=stt.nextToken();
String det=commonMet();
System.out.println("DBSTR===== "+dbst);
if(dbst.equals("Wpin"))
{
    status="error";
    addActionError("Wrong OTP or Pin...Please try
again..");
}
else
{
    status="success";
    System.out.println("Came Success.....");
}
//
url="http://" + det.split("\\=")[0] + ":" + det.split("\\=")[1] + "/EcomClient/successtra
ns?statusvalue="+dbst.trim()+"&sessionId="+sessionId+"&transamt="+amt+"&tknI
d="+tkID.trim()+"&pbId="+productid.trim();

DbQueries dbq=new DbQueries();
PreparedStatement pr=null;
ResultSet rs;
try
{
    String email="";
    pr=(PreparedStatement)
dbq.con.prepareStatement("select mailid from userinfo where accountno=?");
    pr.setString(1, firstbankholdernames);
    rs=pr.executeQuery();
    if(rs.next())
    {
        email=rs.getString("mailid");
    }

    url="http://" + det.split("\\=")[0] + ":" + det.split("\\=")[1] + "/QuantCloud/Bankrespo
nse?mail="+email;
}
catch(Exception e)

```

```

        {
            e.printStackTrace();
        }
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
return status;
}

public String commonMet()
{
    String servername="",portnum="";
    try
    {
        Properties prop=new Properties();
        FileInputStream fis=new
FileInputStream("webapps/BankAdmin/common.properties");
        prop.load(fis);
        servername=prop.getProperty("distserver");
        portnum=prop.getProperty("distport");
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return servername+"="+portnum;
}
}

```

```

package com.banklogic;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.io.Serializable;
public class XIApi implements Serializable,DataMapInterface
{
    BankLogin banklogin;
    public XIApi(String filePath,BankLogin banklogin)
    {
        try
        {
            this.banklogin=banklogin;
            readLines(filePath);
        }
    }
}

```

```

        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
    public void readLines(String filePath)
    {
        try
        {
            FileInputStream fstream1 = new FileInputStream(filePath);
            BufferedReader br1 = new BufferedReader(new
InputStreamReader(fstream1));
            String strLine1;
            int m1=0;
            File ff=new File("webapps/BankAdmin/citynames.txt");
            ff.delete();
            FileOutputStream fos=new
FileOutputStream("webapps/BankAdmin/citynames.txt",true);
            while ((strLine1 = br1.readLine()) != null)
            {
                if(strLine1!="")
                {
                    if(strLine1.contains(","))
                    {
                        if((m1==0))
                        {
                            m1++;
                        }
                        else
                        if(m1>=1)
                        {
                            String dataArray[]=strLine1.split("\\,");
                            if(!pinodelistmap.containsKey(dataArray[0]))
                            {
                                pinodelistmap.put(dataArray[0].trim(),dataArray[1].trim()+"*"+dataArray[2].trim()+"*"+d
ataArray[3].trim());
                                if(m1==1)
                                {
                                    fos.write(dataArray[0].trim().getBytes());
                                }
                                else
                                {
                                    fos.write(("\\n"+dataArray[0].trim()).getBytes());
                                }
                                m1++;
                            }
                        }
                    }
                }
            }
        }
    }

```

```

    }
    }
    fos.close();
}
catch(Exception e)
{
    e.printStackTrace();
}
}
}

```

```

package com.bankwebser;

```

```

public class BankStatementService {
    public String getamount(String userNam)
    {
        String holNam=userNam;
        DbQueries db=new DbQueries();
        String accoutdetails="";
        try
        {
            String holderno=db.getaccountno(holNam);
            accoutdetails=db.statement(holderno);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        return accoutdetails;
    }
    public String viewstatement(String userNam)
    {
        String holNam=userNam;
        DbQueries db=new DbQueries();
        String accoutdetails="";
        try
        {
            String holderno=db.getaccountno(holNam);
            accoutdetails=db.statementdetails(holderno);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        return accoutdetails;
    }
}

```

```

package com.bankwebser;
@javax.jws.WebService(targetNamespace = "http://bankwebser.com/",
serviceName = "BankStatementServiceService", portName =
"BankStatementServicePort", wsdlLocation = "WEB-
INF/wsdl/BankStatementServiceService.wsdl")
public class BankStatementServiceDelegate {
    com.bankwebser.BankStatementService bankStatementService = new
com.bankwebser.BankStatementService();
    public String getamount(String userNam) {
        return bankStatementService.getamount(userNam);
    }
}

```

```

package com.bankwebser;

import java.io.FileInputStream;
import java.net.InetAddress;
import java.util.Properties;
import java.util.Random;

import javax.servlet.http.HttpServletRequest;

public class BankWebService
{
    public String setPayment(String mailid,String payment)
    {
        System.out.println("Request to Bank Webserver");
        String url="";
        try
        {
            Random r=new Random();
            //-----
            char[] map1 = new char[68];

            int i = 0;
            for(char c = 'A'; c <= 'Z'; c++)
            {
                map1[i++] = c;
            }
            for(char c = 'a'; c <= 'z'; c++)
            {
                map1[i++] = c;
            }
            for (char c = '0'; c <= '9'; c++)
            {
                map1[i++] = c;
            }
            map1[i++] = '+';
            map1[i++] = '/';

```

```

map1[i++] = '$';
map1[i++] = '#';
map1[i++] = '&';
map1[i++] = '*';

String p1=String.valueOf(map1[r.nextInt(67)]);
String p2=String.valueOf(map1[r.nextInt(67)]);
String p3=String.valueOf(map1[r.nextInt(67)]);
String p4=String.valueOf(map1[r.nextInt(67)]);
String p5=String.valueOf(map1[r.nextInt(67)]);
String p6=String.valueOf(map1[r.nextInt(67)]);
String p7=String.valueOf(map1[r.nextInt(67)]);
String p8=String.valueOf(map1[r.nextInt(67)]);
String onetypassword=p1+p2+p3+p4+p5+p6+p7+p8;
//-----

String str=commonMet();
String spl[]=str.split("\\=");
String to=mailid.trim();
String from = spl[0].trim();
String subject = "MAIL ALERT - PASSCODE";
String message="Your OPASS :"+onetypassword+"";
String host = "smtp.gmail.com";
// Mail msend = new
Mail(host,from,spl[1].trim(),from,to,subject,message);

DbQueries db=new DbQueries();
db.insertOpassUser(mailid,onetypassword,payment);

String ipaddr=InetAddress.getLocalHost().getHostAddress();
String port=new CommonAccess().getPort();

url="http://"+ipaddr+":"+port+"/BankAdmin/setuserpayment.jsp";
System.out.println("url-----"+url);
}
catch(Exception e)
{
    e.printStackTrace();
}
return url;
}

public String commonMet()
{
    String accmail="",mailpassword="";
    try
    {
        Properties prop=new Properties();
        FileInputStream fis=new
FileInputStream("webapps/BankWebServer/Mail.properties");

```

```

        prop.load(fis);
        accmail=prop.getProperty("mailid");
        mailpassword=prop.getProperty("mailpassword");
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return accmail+"="+mailpassword;
}
}

```

```

package com.bankwebser;
import java.io.FileInputStream;
import java.net.InetAddress;
import java.util.Properties;
import java.util.Random;
import javax.servlet.http.HttpServletRequest;
@javax.jws.WebService(targetNamespace = "http://bankwebser.com/",
serviceName = "BankWebServiceService", portName = "BankWebServicePort",
wsdlLocation = "WEB-INF/wsdl/BankWebServiceService.wsdl")
public class BankWebServiceDelegate {
    com.bankwebser.BankWebService bankWebService = new
com.bankwebser.BankWebService();
    public String setPayment(String mailid, String payment) {
        return bankWebService.setPayment(mailid, payment);
    }
    public String commonMet() {
        return bankWebService.commonMet();
    }
}
}

```

```

package com.bankwebser;

import java.io.FileInputStream;
import java.util.Properties;

public class CommonAccess
{
    public String getPort()
    {
        String port="";
        try
        {
            Properties prop=new Properties();
            FileInputStream fis=new
FileInputStream("webapps/BankWebServer/commondata.properties");
            prop.load(fis);

```

```

        port=prop.getProperty("portnumber").trim();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return port;
}
}

```

```

package com.bankwebser;

```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

```

```

import javax.servlet.ServletContext;
import org.apache.struts2.ServletActionContext;
import com.mysql.jdbc.exceptions.MySQLIntegrityConstraintViolationException;

```

```

public class DbQueries
{

```

```

    String qu="";
    Connection con=null;
    PreparedStatement pst=null;

```

```

    public DbQueries()
    {

```

```

        try
        {
            Class.forName("com.mysql.jdbc.Driver").newInstance();

```

```

            con=DriverManager.getConnection("jdbc:mysql://localhost/bankadmindb","root","root");

```

```

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

```

```

    public void insertOpassUser(String mailid,String opass,String transamount)
    {

```

```

        try
        {
            String accnt="";

```

```

        pst=con.prepareStatement("select accountno from userinfo
where mailid=?");
        pst.setString(1,mailid);
        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {
            acctnt=rs.getString(1);
        }
        pst=con.prepareStatement("update useraccountinfo set
opass=?,transferamount=? where accountno=?");
        pst.setString(1,opass);
        pst.setString(2,transamount);
        pst.setString(3,acctnt);
        pst.executeUpdate();
        pst.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
public String getaccountno(String holdrName)
{
    String holderaccno="";
    try
    {
        String quer="select accountno from userinfo where
accountholdername=?";
        pst=con.prepareStatement(quer);
        pst.setString(1,holdrName);
        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {
            holderaccno=rs.getString(1).toString().trim();
            System.out.println("holderaccno====="+holderaccno);
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return holderaccno;
}
public String statement(String uld)
{
    String str="";
    String amount="";
    try
    {
        qu="SELECT userinfo.*,useraccountinfo.amount"

```

```

        +" FROM userinfo"
        +" INNER JOIN useraccountinfo"
        +" ON userinfo.accountno=useraccountinfo.accountno"
        +" where userinfo.accountno=?";

        pst=con.prepareStatement(qu);
        pst.setString(1,uld);
        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {
            amount=rs.getString(9).toString().trim();

            //str=uld+"&" +rs.getString(2)+"&" +rs.getString(9)+"&" +rs.getString(3)+"&" +r
            s.getString(4)+"&" +rs.getString(5)+"&" +rs.getString(6)+"&" +rs.getString(7)+"&" +rs.
            getString(8);
        }
        pst.close();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return amount;
}
public String statementdetails(String accno)
{
    String statement="";
    try
    {
        String qu="select accholdername,time,productid,productamt,tokenid
from userinfo where =?";
        pst=con.prepareStatement(qu);
        pst.setString(1,accno);
        ResultSet rs=pst.executeQuery();
        while(rs.next())
        {

            statement=rs.getString(1)+"@" +rs.getString(2)+"@" +rs.getString(3)+"@" +rs
            .getString(4)+"@" +rs.getString(5);
        }
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return statement;
}
}

```

```

package com.bankwebser;

import java.util.*;
import javax.mail.Session;
import javax.mail.Message.RecipientType;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import com.sun.mail.smtp.SMTPSSLTransport;

public class Mail
{
    public Mail(String host,String userid,String password,String from,String
to,String subject,String message)
    {
        try
        {
            Properties props = System.getProperties();
            props.put("mail.smtp.starttls.enable", "true");
            props.put("mail.smtp.host", host);
            props.setProperty("mail.transport.protocol", "smtps");
            props.put("mail.smtp.user", userid);
            props.put("mail.smtp.password", password);
            props.put("mail.smtp.port", "465");
            props.put("mail.smtps.auth", "true");

            Session session = Session.getDefaultInstance(props, null);
            MimeMessage message = new MimeMessage(session);
            InternetAddress fromAddress = null;
            InternetAddress toAddress = null;
            try
            {
                System.out.println("Address of the sender==="+from);
                System.out.println("Address of the reciever==="+to);
                fromAddress = new InternetAddress(from);
                toAddress = new InternetAddress(to);
            }
            catch(AddressException e)
            {
                e.printStackTrace();
            }
            System.out.println("Content of the title displayed"+subject);
            System.out.println("Content of the message displayed"+message);
            message.setFrom(fromAddress);
            message.setRecipient(RecipientType.TO, toAddress);
            message.setSubject(subject);
            message.setText(message);
            SMTPSSLTransport transport
=(SMTPSSLTransport)session.getTransport("smtps");
            transport.connect(host, userid, password);
        }
    }
}

```

```

        transport.sendMessage(message, message.getAllRecipients());
        transport.close();
        System.out.println("Mail Sent Successfully");
    }
    catch(Exception e1)
    {
        e1.printStackTrace();
    }
}
}

```

package DB;

```

import java.sql.Connection;
import java.sql.DriverManager;

```

```

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

```

```

public class DBconnection implements ServletContextListener
{
    static Connection connection = null;

    public void contextDestroyed(ServletContextEvent arg0)
    {
        try
        {
            System.out.println("-----CrowdFunding connection closed
successfully-----");

            connection.close();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    public void contextInitialized(ServletContextEvent arg0)
    {
        ServletContext context = arg0.getServletContext();
        try
        {
            Class.forName("com.mysql.jdbc.Driver");

```

```

        connection =
DriverManager.getConnection("jdbc:mysql://localhost/voiceofcharity", "root",
"root");
        context.setAttribute("connection", connection);
        System.out.println("-----CrowdFunding connection opened
successfully-----");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

public Connection getConnection()
{
    return connection;
}
}

package logic;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.Statement;
import java.util.Properties;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.http.servlet.HttpServlet;

    public class Cancal extends HttpServlet {

        public void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            String email = request.getParameter("email");

            try {
                System.out.println("Email---->" + email);

```

```

        String s1 = "Declined";
        System.out.println(s1);
        String q = "you are not eligible";
        Connection connection = (Connection)
getServletContext()
                                .getAttribute("connection");

        System.out.println("connection success");

        Statement st = connection.createStatement();
        int i = st.executeUpdate("update donorcard set
status1='' + s1
                                + '' where Email='' + email + ''");
        File pf = new File("");
        String proppath = pf.getAbsolutePath();
        String filepath = proppath
                                +
        "\\webapps\\CrowdFunding\\Properties\\MailProperties.properties";
        InputStream pfis =
this.getClass().getResourceAsStream(
                                "MailProperties.properties");
        Properties prop = new Properties();
        prop.load(pfis);
        String empuserid = prop.getProperty("adminid");
        String empmailpassword =
prop.getProperty("password");
        String subject = "Charity Application Status";
        String text = "Hello ... \n Your application has been " +
s1
                                + " \t Thanks & Regards \t Cloud OTP
Service";

        Mail sme = new Mail(empuserid, email , subject, text,
empuserid,empmailpassword);
        request.setAttribute("msg", "User is declined");
        RequestDispatcher rd = request
                                .getRequestDispatcher("ddetails.jsp");
        rd.include(request, response);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

}

```

package logic;

```

import java.io.IOException;
import java.io.PrintWriter;

```

```

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.http.servlet.HttpServlet;

public class Chart extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        ArrayList<String> al = new ArrayList<String>();
        String id = request.getParameter("id");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String email = request.getParameter("email");
        try {
            HttpSession session = request.getSession(false);
            Connection connection = (Connection) getServletContext()
                .getAttribute("connection");
            System.out.println("connection success");
            Statement st = connection.createStatement();
            ResultSet rs = st
                .executeQuery("select * from donorcateg where email='"+
email);
            System.out.println(rs.toString());

            while (rs.next()) {
                String email1 = rs.getString("email");
                String category = rs.getString("category");

                al.add(email1 + "@" + category);
            }
            System.out.println("Array List === " + al);
            request.setAttribute("al", al);

            RequestDispatcher rd =
request.getRequestDispatcher("charts.jsp");
            rd.forward(request, response);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        }

        out.close();
    }
}

package logic;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.http.servlet.HttpServlet;

public class DonationGraph extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        //String id = request.getParameter("id");
        ArrayList<String> al = new ArrayList<String>();

        try {
            Connection connection = (Connection) getServletContext()
                .getAttribute("connection");
            System.out.println("connection success");
            Statement st = connection.createStatement();
            ResultSet rs = st
                .executeQuery("select status from categories
group by status");
            System.out.println(rs.toString());
            while (rs.next()) {
                String status = rs.getString("status");

                //      al.add(total_time + "@" + category);

```

```

        }
        System.out.println("Array List === " + al);
        request.setAttribute("al", al);

        RequestDispatcher rd =
request.getRequestDispatcher("charts.jsp");
        rd.forward(request, response);

        } catch (Exception e) {
            e.printStackTrace();
        }

        out.close();
    }
}

```

```
package logic;
```

```

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import java.util.Random;

```

```

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

```

```

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;

```

```
import logic.PythonCall;
```

```
import com.http.servlet.HttpServlet;
```

```
public class GenerateID extends HttpServlet {
```

```

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

```

```

response.setContentType("text/html");
PrintWriter out = response.getWriter();

String arr[];
ArrayList<String> list = new ArrayList<String>();
String PanCard = request.getParameter("pancardimage");
String email = request.getParameter("email1");
String Pan = request.getParameter("pan");
String reason = request.getParameter("reason");

// HttpSession httpsession = request.getSession(true);
// String Email = (String) httpsession.getAttribute("email1");

try {
    System.out.println("======" + email + "Pan.jpg");
    File f = new File("webapps//CrowdFundingg//PanRecipient//");
    PythonCall callPython = new PythonCall();
    List<String> respo = callPython.executeMultiPartRequest(
        "http://localhost:5001/upload", new File(f, email
            + "pan.jpg"));
    String result1 = respo.toString();
    System.out.println("----->"
        + result1.replaceAll("\n\n", "").trim());

    JSONArray jaArray = (JSONArray) new
JSONParser().parse(result1);
    JSONObject jsonObject = (JSONObject) jaArray.get(0);
    String data = String.valueOf(jsonObject.get("msg"));
    System.out.println("===-----==" + data.trim());

    String temp = data.replaceAll("[\t\n\r]+", " ");
    System.out.println("-----temp----->" + temp.trim());

    arr = temp.trim().split(" ");

    for (String str : arr) {
        list.add(str);
    }
    System.out.println("-----list----->>>" + list);

    System.out.println("----->" + Pan);
    boolean at = list.contains(Pan);
    System.out.println("======" + at);

    if (at == true) {

        System.out.println("Email---->" + email);

        String s1 = createid();

```

```

        System.out.println(s1);
        String q = "you are eligible";
        Connection connection = (Connection)
getServletContext()
        .getAttribute("connection");

        System.out.println("connection success");

        Statement st = connection.createStatement();
        int i = st.executeUpdate("update kycdetails set
status1="" + s1
        + "" where email1="" + email + """);
        File pf = new File("");
        String proppath = pf.getAbsolutePath();
        String filepath = proppath
        +
        "\\webapps\\CrowdFundingg\\Properties\\MailProperties.properties";
        InputStream pfis =
this.getClass().getResourceAsStream(
        "MailProperties.properties");
        Properties prop = new Properties();
        prop.load(pfis);
        String empuserid = prop.getProperty("adminid");
        String empmailpassword =
prop.getProperty("password");
        String subject = "Charity Application Status";
        String text = "Hello ... \n Your Login Password is " + s1
        + " \t Thanks & Regards \t Cloud OTP
Service";
        Mail sme = new Mail(empuserid, email , subject, text,
empuserid,empmailpassword);

        request.setAttribute("msg", "User approved
Successfully");

        RequestDispatcher rd = request
        .getRequestDispatcher("rdetails.jsp");
        rd.include(request, response);
    } else {

        System.out.println("=====*****=====");

        RequestDispatcher rd =
request.getRequestDispatcher("Cancel");
        rd.include(request, response);
    }

    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

    }

    public String createid() {
        int counter;
        Random rnum = new Random();
        System.out.println("Random Numbers:");
        System.out.println("*****");
        int s = rnum.nextInt(2000);

        String a = "UID_" + s;
        return a;
    }
}

package logic;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import java.util.Random;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;

import com.http.servlet.HttpsServlet;

public class GenerateID1 extends HttpsServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String arr[];

```

```

ArrayList<String> list = new ArrayList<String>();
String PanCard = request.getParameter("pancardimage");
String email = request.getParameter("email");
String Pan = request.getParameter("pan");
String reason = request.getParameter("reason");

try {
    System.out.println("=====" + email + "Pan.jpg");
    File f = new File("webapps//CrowdFundingg//PanDonor//");
    PythonCall callPython = new PythonCall();
    List<String> respo = callPython.executeMultiPartRequest(
        "http://localhost:5001/upload", new File(f, email
            + "pan.jpg"));
    String result1 = respo.toString();
    System.out.println("----->"
        + result1.replaceAll("\n\n", "").trim());

    JSONArray jaArray = (JSONArray) new
JSONParser().parse(result1);
    JSONObject jsonObject = (JSONObject) jaArray.get(0);
    String data = String.valueOf(jsonObject.get("msg"));
    System.out.println("===-----==" + data.trim());

    String temp = data.replaceAll("[\t\n\r]+", " ");
    System.out.println("-----temp----->" + temp.trim());

    arr = temp.trim().split(" ");

    for (String str : arr) {
        list.add(str);
    }
    System.out.println("-----list----->>>" + list);

    System.out.println("-----pan card----->" + Pan);
    boolean at = list.contains(Pan);
    System.out.println("=====" + at);

    if (at == true) {
        System.out.println("Email---->" + email);

        String s1 = createid();
        System.out.println(s1);
        String q = "you are eligible";
        Connection connection = (Connection) getServletContext()
            .getAttribute("connection");

        System.out.println("connection success");

        Statement st = connection.createStatement();

```

```

        int i = st.executeUpdate("update donorcard set status1=" + s1 + "
where email=" + email
                                + "");

        File pf = new File("");
        String proppath = pf.getAbsolutePath();
        String filepath = proppath
                                +
        "\\webapps\\CrowdFundingg\\Properties\\MailProperties.properties";
        InputStream pfis = this.getClass().getResourceAsStream(
                                "MailProperties.properties");
        Properties prop = new Properties();
        prop.load(pfis);
        String empuserid = prop.getProperty("adminid");
        String empmailpassword = prop.getProperty("password");
        String subject = "Charity Application Status";
        String text = "Hello ... \n Your Login Password is " + s1
                                + " \t Thanks & Regards \t Cloud OTP Service";

        Mail sme = new Mail(empuserid, email, subject, text, empuserid,
empmailpassword);

        request.setAttribute("msg", "Donor approved Successfully");
        RequestDispatcher rd =
request.getRequestDispatcher("ddetails.jsp");
        rd.include(request, response);
    }else{
        System.out.println("=====*****=====");

        RequestDispatcher rd = request.getRequestDispatcher("Canca");
        rd.include(request, response);
    }
}

catch (Exception e) {

    e.printStackTrace();
}

}

public String createid() {
    int counter;
    Random rnum = new Random();
    System.out.println("Random Numbers:");
    System.out.println("*****");
    int s = rnum.nextInt(2000);

    String a = "DID_" + s ;
    return a;
}

}

```

```

package logic;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.http.servlet.HttpServlet;

public class loginform extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        PrintWriter out = response.getWriter();
        if (email.equals("admin@gmail.com") && password.equals("admin"))
        {
            RequestDispatcher rd =
request.getRequestDispatcher("Options.jsp");
            rd.forward(request, response);
        } else {
            request.setAttribute("msg", "Invalid Admin Email or
Password");
            RequestDispatcher rd =
request.getRequestDispatcher("login.jsp");
            rd.forward(request, response);
        }
        out.close();
    }
}

```

```

package logic;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;

```

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import com.http.servlet.HttpsServlet;

public class Logout extends HttpsServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession(false);
        session.invalidate();
        response.sendRedirect("index.jsp");
        out.close();
    }
}

package logic;

import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.Message.RecipientType;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import com.sun.mail.smtp.SMTPSSLTransport;

public class Mail {
    public Mail(String from, String to, String subject, String text,
        String mailusername, String mailpassword) {
        String host = "smtp.gmail.com";
        try {
            Properties props = System.getProperties();
            props.put("mail.smtp.starttls.enable", "true");
            props.put("mail.smtp.host", host);
            props.setProperty("mail.transport.protocol", "smtps");
            props.put("mail.smtp.user", mailusername);
            props.put("mail.smtp.password", mailpassword);
            props.put("mail.smtp.port", "465");
            props.put("mail.smtps.auth", "true");
            Session session = Session.getDefaultInstance(props, null);

```

```

        MimeMessage message = new MimeMessage(session);
        InternetAddress fromAddress = null;
        InternetAddress toAddress = null;
        try {
            fromAddress = new InternetAddress(from);
            toAddress = new InternetAddress(to);
        } catch (AddressException e) {
            e.printStackTrace();
        }
        message.setFrom(fromAddress);
        message.setRecipient(RecipientType.TO, toAddress);
        message.setSubject(subject);
        message.setText(text);
        SMTPSSLTransport transport = (SMTPSSLTransport)
session.getTransport("smtps");
        Transport transport1 = session.getTransport("smtps");
        transport.connect(host, mailusername, mailpassword);
        transport.sendMessage(message,
message.getAllRecipients());
        transport.close();
    } catch (MessagingException e) {
        e.printStackTrace();
    }
}
}

```

```

package logic;

```

```

import java.io.BufferedReader;
import java.io.File;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

```

```

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.mime.MultipartEntity;
import org.apache.http.entity.mime.content.FileBody;
import org.apache.http.impl.client.DefaultHttpClient;

```

```

@SuppressWarnings("deprecation")
public class PythonCall {

```

```

    @SuppressWarnings("deprecation")
    public List<String> executeMultiPartRequest(String urlString, File file)
throws Exception

```

```

{
    List<String> respons = new ArrayList<String>();
    HttpClient client = new DefaultHttpClient() ;
    HttpPost postRequest = new HttpPost (urlString) ;
    try
    {
        //Set various attributes
        MultipartEntity multiPartEntity = new MultipartEntity() ;

//        multiPartEntity.addPart("name", new StringBody(fileName)) ;

        FileBody fileBody = new FileBody(file, "application/octet-stream") ;
        //Prepare payload
        multiPartEntity.addPart("file", fileBody);
//        multiPartEntity.addPart("email", new StringBody(email));
//        multiPartEntity.addPart("imagename", new StringBody(email));
        //Set to request body
        postRequest.setEntity(multiPartEntity) ;
        //Send request
        HttpResponse response = client.execute(postRequest) ;

        System.out.println("-Response-----
"+response.getStatusLine().getStatusCode());
        //Verify response if any
        if (response != null)
        {
            BufferedReader reader = new BufferedReader(new InputStreamReader(
                response.getEntity().getContent()));
            String line = null;
            while ((line = reader.readLine()) != null) {
                respons.add(line);
            }
            reader.close();

            return respons;
        }else{
            return null;
        }
    }
    catch (Exception ex)
    {
        return null;
    }
}
}

```

package logic;

```

import java.io.File;
import java.io.FileWriter;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;

public class XmlCreation
{

    public XmlCreation()
    {
    }

    public void createfile(String folder)
    {
        File fp = new File((new
StringBuilder("webapps/EfficientRegularLanguage/").append(folder).toString());
        if(!fp.exists())
            fp.mkdirs();
    }

    public void updatepolicy(String name, String email,String pm,String
Student,String sse,String Professional,String se,String Others)
    {
        try
        {
            File file = new File((new
StringBuilder("webapps/EfficientRegularLanguage/PolicyFiles/").append(email).a
ppend(".xml").toString());
            // if(file.exists())
            // {
            //     SAXBuilder sb = new SAXBuilder();
            //     File fpp = new File((new
StringBuilder("webapps/CCDataOwners/PolicyFiles/").append(email).append(".x
ml").toString());
            //     Document doc = sb.build(fpp);
            //     Element root = doc.getRootElement();
            //     java.util.List fdetail = root.getChildren();
            //     Element comment = new Element("Policy");
            //     comment.addContent((new Element("username")).setText(name));
            //     comment.addContent((new Element("role")).setText(pm));
            //     comment.addContent((new
Element("permission")).setText(project_manager));
            //     comment.addContent((new Element("role")).setText(sse));
            //     comment.addContent((new
Element("permission")).setText(srsoft_eng));
            //     comment.addContent((new Element("role")).setText(se));

```

```

//      comment.addContent((new Element("permission")).setText(soft_eng));
//      comment.addContent((new Element("role")).setText(hrm));
//      comment.addContent((new
Element("permission")).setText(hr_manager));
//      comment.addContent((new Element("role")).setText(tl));
//      comment.addContent((new
Element("permission")).setText(team_lead));
//      comment.addContent((new Element("role")).setText(at));
//      comment.addContent((new
Element("permission")).setText(assoc_trainee));
//
//      doc.getRootElement().addContent(comment);
//      XMLOutputter out = new XMLOutputter();
//      out.setFormat(Format.getPrettyFormat());
//      out.output(doc, new FileWriter((new
StringBuilder("webapps/CCDataOwners/PolicyFiles/")).append(email).append(".x
ml").toString()));
//      } else
//      {
//          Element root = new Element("Policies");
//          Document doc = new Document(root);
//          doc.setRootElement(root);
//          Element comment = new Element("Policy");
//          comment.addContent((new Element("username")).setText(name));
//          comment.addContent((new Element("role")).setText(pm));
//          comment.addContent((new Element("permission")).setText(Student));
//          comment.addContent((new Element("role")).setText(sse));
//          comment.addContent((new
Element("permission")).setText(Professional));
//          comment.addContent((new Element("role")).setText(se));
//          comment.addContent((new Element("permission")).setText(Others));
//          doc.getRootElement().addContent(comment);
//          XMLOutputter out = new XMLOutputter();
//          out.setFormat(Format.getPrettyFormat());
//          out.output(doc, new FileWriter((new
StringBuilder("webapps/EfficientRegularLanguage/PolicyFiles/")).append(email).a
ppend(".xml").toString()));
//      }
//  }
//  catch(Exception e)
//  {
//      e.printStackTrace();
//  }
//  }
}

```

