1. INTRODUCTION

1.1 Introduction

The **Organ Donation Management System (ODMS)** is developed mainly for general hospitals (GH), clinics and other health centers to manage the donor registration and user maintenance.

The public can retrieve information about organ donation through this desktop application. People who are interested in organ donation can register themselves through this system.

The application will be processed by the administrator and each donor will receive feedback about their application status. The donor can visit specified medical centers to perform a medical check-up or can ask for a medical assistance at home.

Only administrator has the authority and privileges to print organ list report and total donation report according to district from this system. An analysis study has been done based on the current manual system and all the problems statements and requirements have been identified.

In addition, the improvement part of this system is to help the administrators to easily retrieve the donor's details. Other than that, it also supports the data integrity for each and every change which is done on to the system. This system also assures the data integrity and helps the management handle the donor's registration more efficiently.

This Online Organ Donation Management System will help to improve the current situation and overcome the problems that arise nowadays.

1.2 Problem Formulation

The current Organ Donation System still using the manual file system which is also known as a simple database. Ledgers and logbooks are wisely used to record the information and evens of the donors. Moreover, the current online system is only for retrieve information about organ donation and donor’s registration. There are fewer functions for administrators where they need to calculate and organized the total registration of donors manually. Rather than that, there is no response or feedback to donors regarding to their registration. This Organ Donation Management System (ODMS) will help the donors, administrators and staff of the organ donation department for a better performance.

1 DESCRIPTION

2.1 Uses:

- To efficiently manage organ donor and recipients profiles.
- Reduce inconsistency in recorded data.
- Help Clinics or hospitals to keep up to date record.
- Archive data for analysis or report generation for human resource management.

2.2 Application:

The ODMS can be implemented in various sectors such as:

- Hospital management system where the record section can keep organ transplant data in secure state.
- Government Agencies to keep vigil eyesight over organ transplantations thus ensuring arrest on illegal tissue trafficking.
- Government schemes to promote organ donation.

3 MODULES

3.1 Home:

In this module, short information about Organ Donation along with the need to be an organ donor is given. It also motivates the donors to voluntarily donate their organs as it is the ultimate humanitarian act of charity.

3.2 Registration:

In this module the registration of both donor and receiver will we carried out. Each user need to fill all the required details correctly will get their unique Id type which they will use to log-in into the system.

3.3 Login:

In this module the receiver can log into his/her profile. The log-in Id will be one provide during the profile registration. In this module admin login is performed where he can search and delete records.

The login module is further divided into three sub modules.

The modules to pop up will be defined by the log-in Id type.

2A: Receiver login:

This module will allow updates to the receiver profile.

The hospitals, admin will have partial control over such profiles.

2B: Admin login:

These profiles will be internally generated in the system.

Such profiles will not have any registration platform.

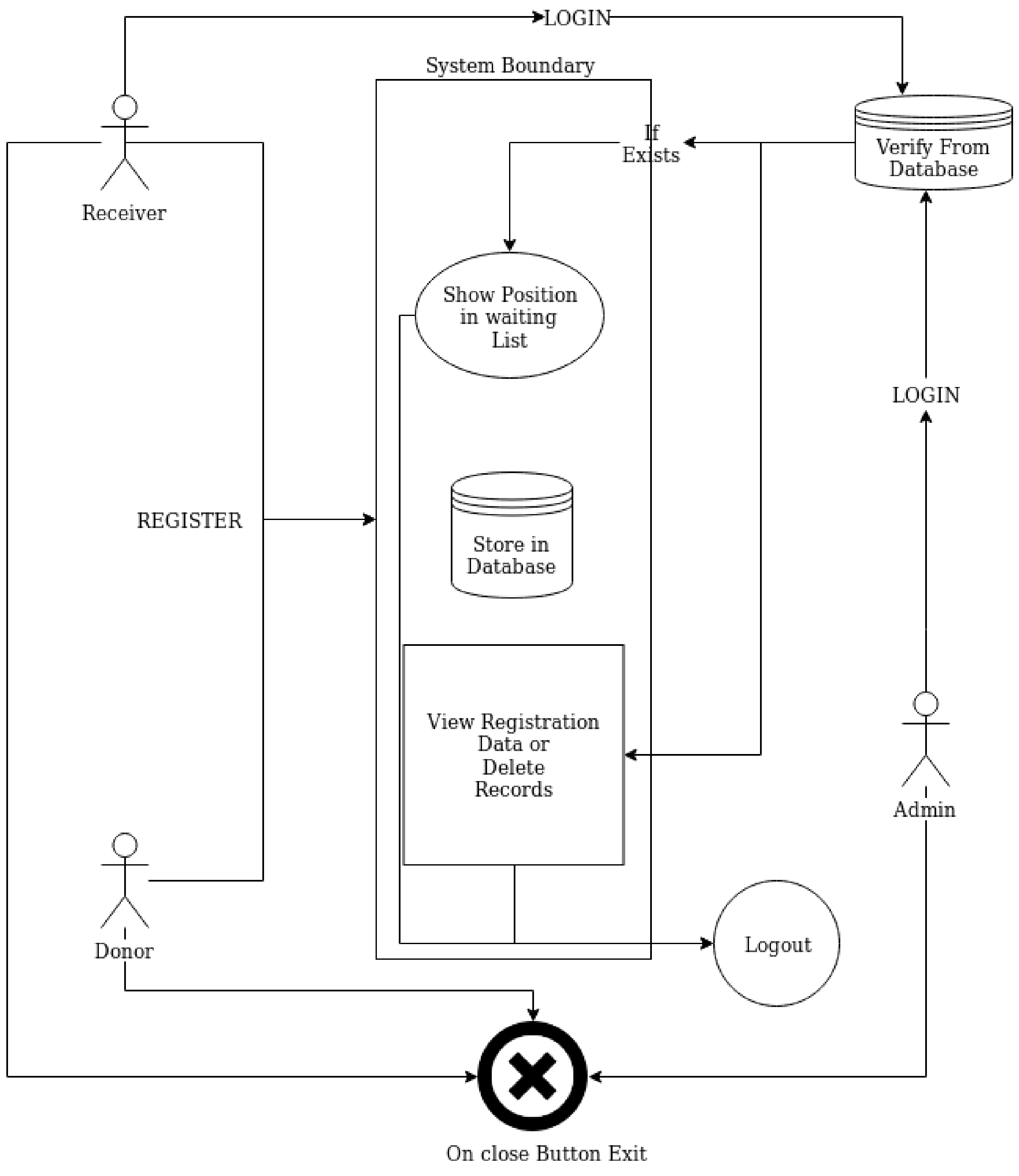
Admin could be hospitals, GC, governing bodies.

3.4 About

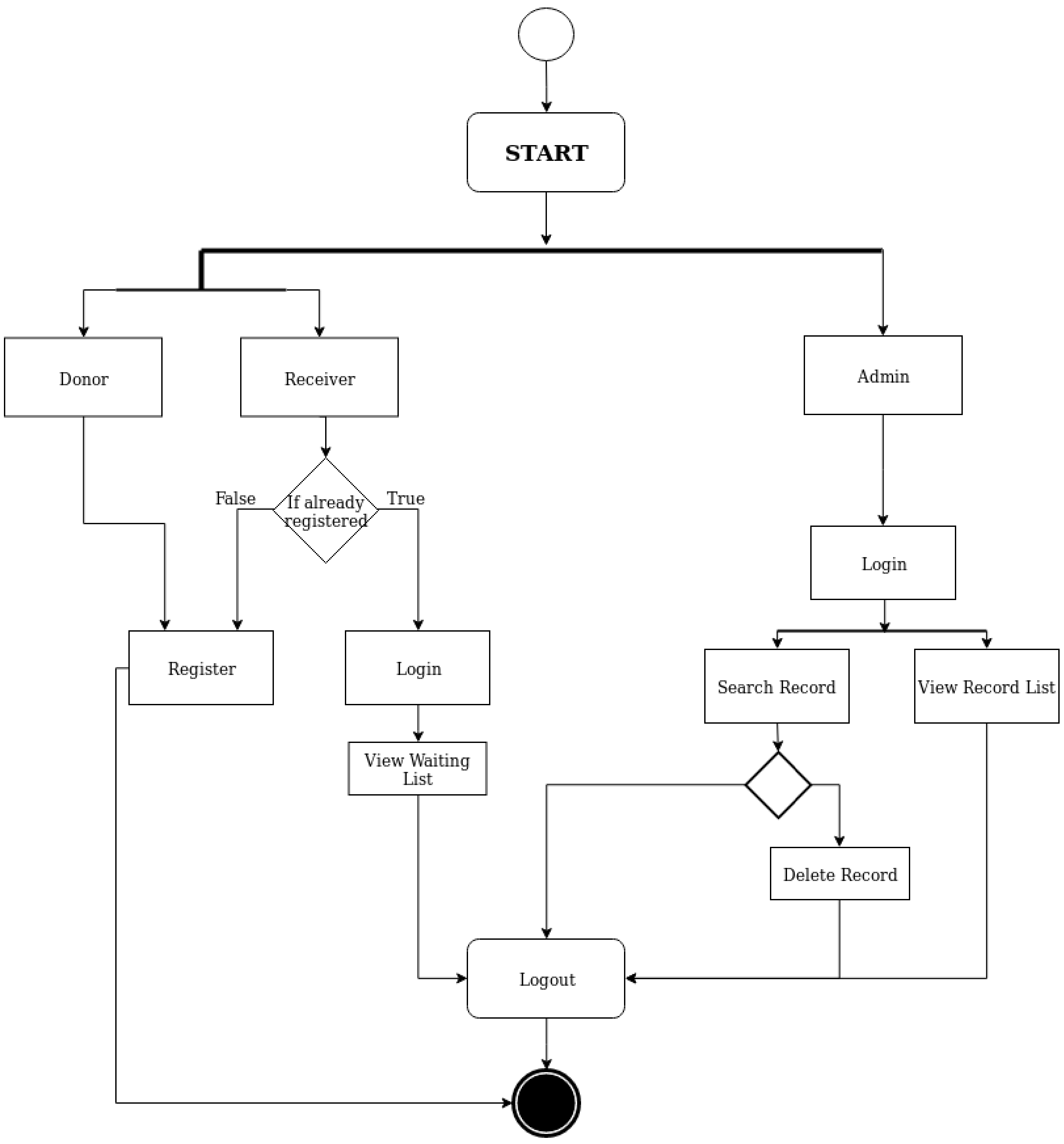
In this module, the name of the project along with the version and copyrights are given.

4 METHODOLOGY

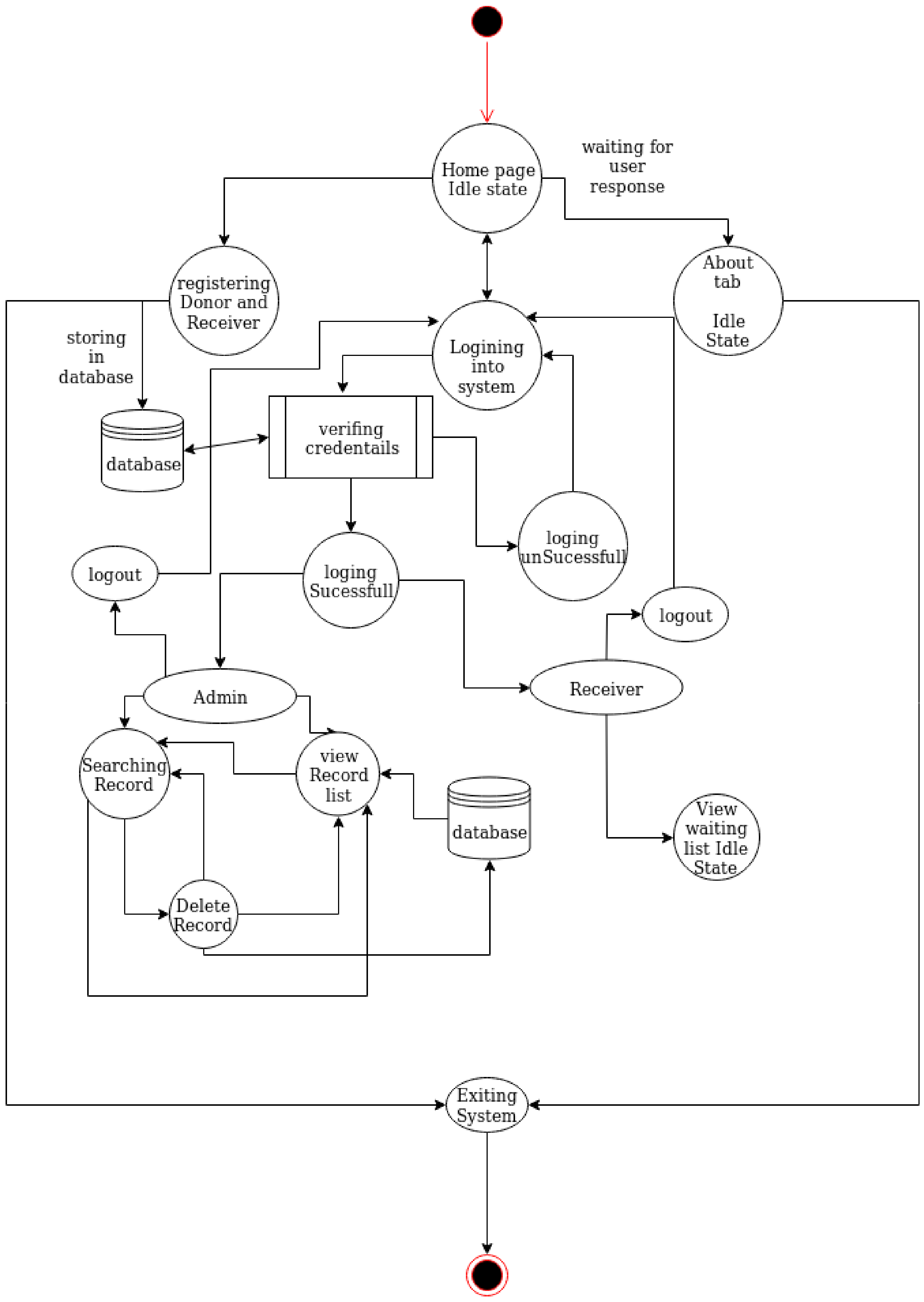
4.1 Use-Case Diagram



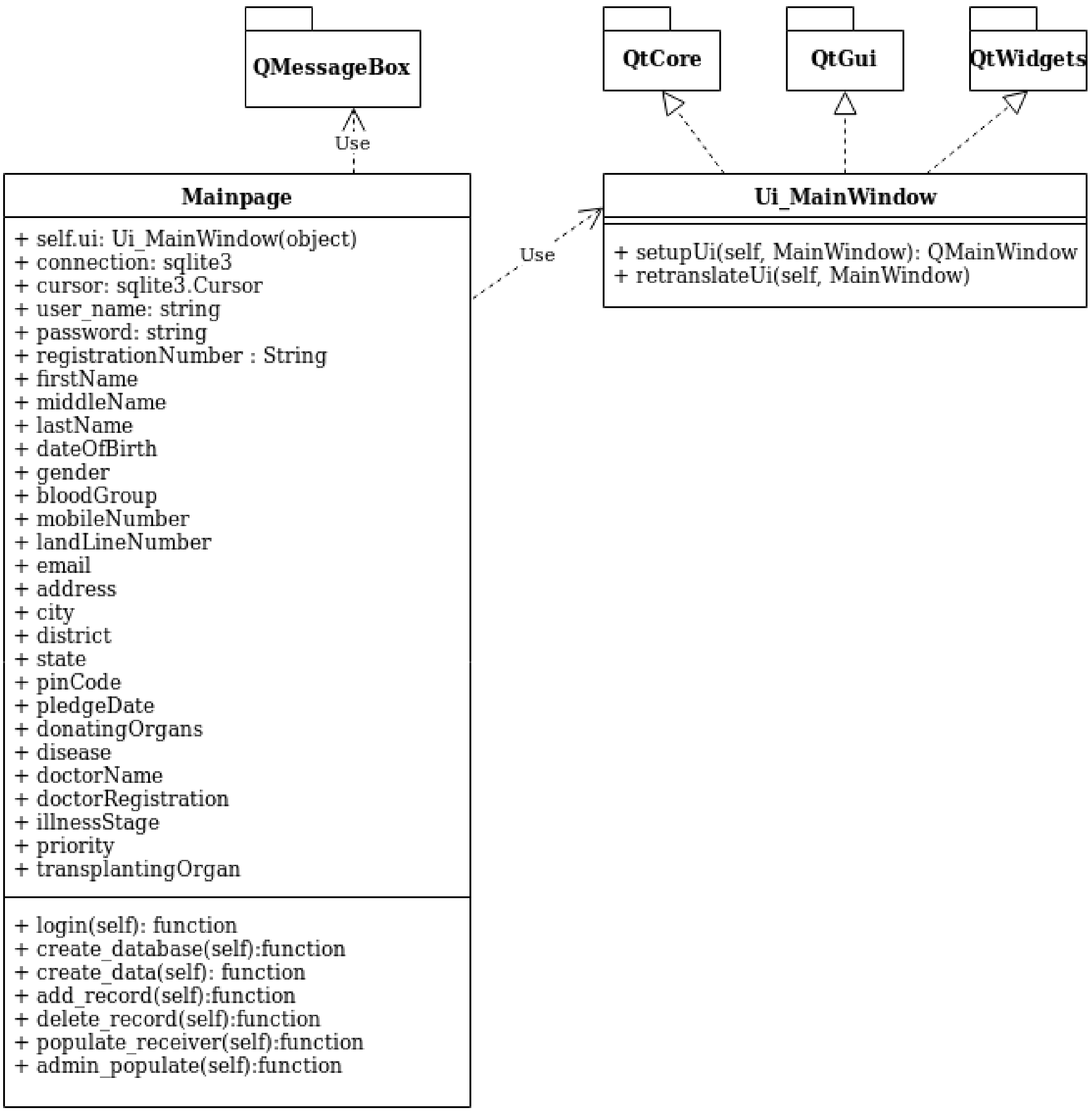
4.2 Activity Diagram



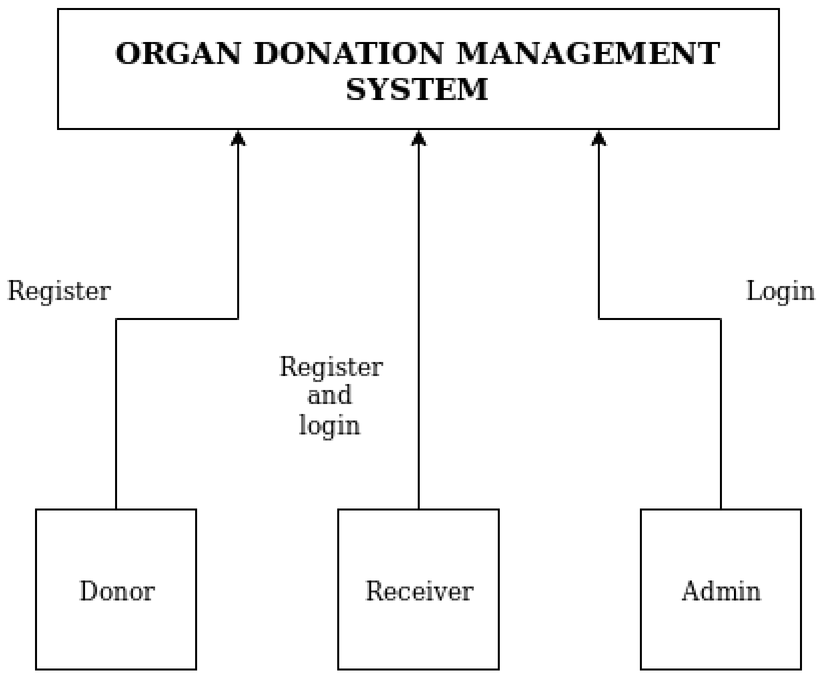
4.3 State Diagram



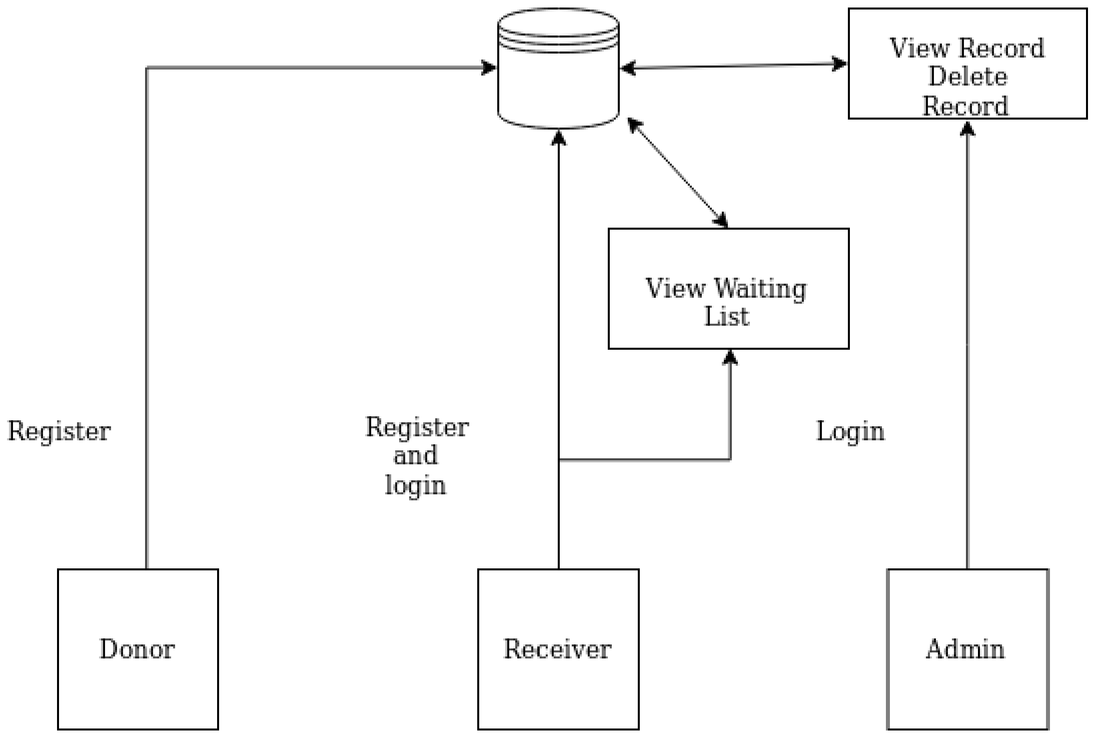
4.4 Class Diagram



4.5 Data Flow Diagram



Level 0



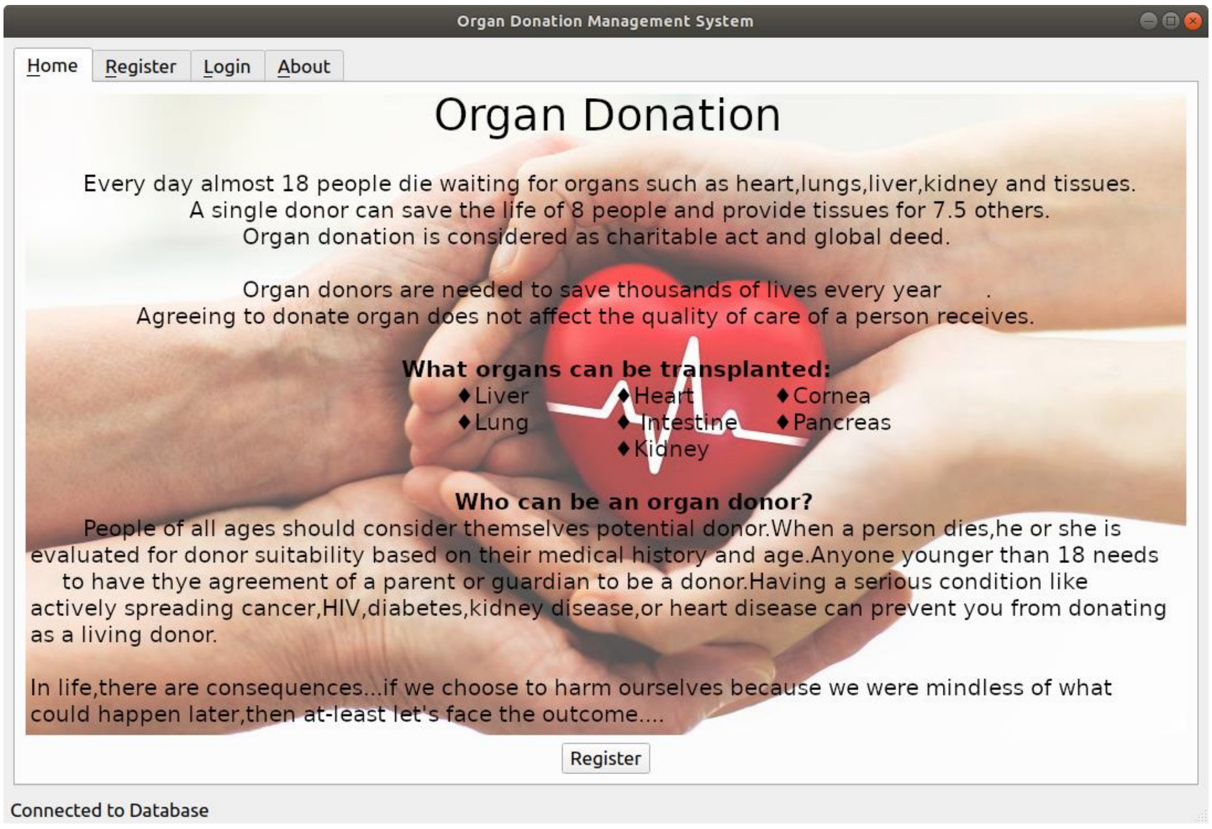
5 CODING

```
import datetime
import re
import sqlite3
import sys

from PyQt5.QtWidgets import QMessageBox
from PyQt5 import QtWidgets, uic
from PyQt5 import QtGui

class Test:
    cursor: sqlite3.Cursor

    def __init__(self):
        self.ui = uic.loadUi('organ.ui')
        self.ui.show()
        self.create_database()
        self.ui.loginButton.clicked.connect(self.login)
        self.ui.homeTabRegisterButton.clicked.connect(self.tab_change)
        self.ui.formSelecter.currentIndexChanged.connect(self.change_form)
        self.ui.registerTabRegisterButton.clicked.connect(self.add_record)
        self.ui.goToLoginPageReceiver.clicked.connect(self.set_login_page)
        self.ui.goToLoginPageAdmin.clicked.connect(self.set_login_page)
        self.ui.deleteRecord.clicked.connect(self.delete_record)

def login(self):
        user_name: str
        password: str
        self.ui.statusNameLabel.setText('')
        self.ui.statusPasswordLabel.setText('')
        try:
            user_name = self.ui.userName.text()
            password = self.ui.userPassword.text()
            if user_name == "" and password == "":
                raise Exception('both')
            if user_name == "":
                raise Exception('user_name')
            if password == "":
                raise Exception('password')
        except Exception as e:
            if str(e) == 'both':
                self.ui.statusNameLabel.setText('* UserName Required')
                self.ui.statusPasswordLabel.setText('* Password Required')
            if str(e) == 'user_name':
                self.ui.statusNameLabel.setText('* UserName Required')
            if str(e) == 'password':
                self.ui.statusPasswordLabel.setText('* Password Required')
        else:
            check = None
            if 'Admin' in user_name:
                self.cursor.execute('select * from loginId where username = ? and password
= ?;',
                                    (user_name, password))
                check = self.cursor.fetchall()
                if check:
                    self.populate_record()
                    self.ui.changeLoginTypeStack.setCurrentIndex(1)

def create_database(self):
        try:
            self.connection = sqlite3.connect('organRecord.db')
            self.cursor = self.connection.cursor()
            self.create_data()
            self.ui.statusbar.showMessage("Connected to Database")
        except Exception as e:
            print(e)
            self.ui.statusbar.showMessage(" Failed to Connect to Database:   Exiting
Application")
            app.exit()

    def create_data(self):
        try:
            query = """CREATE TABLE IF NOT EXISTS receiver(registrationNumber
VARCHAR(5), firstName VARCHAR(20),middleName VARCHAR(20), lastName
VARCHAR(20), dateOfBirth DATE, gender VARCHAR(10), bloodGroup
VARCHAR(3), mobileNumber VARCHAR(15), landLineNumber VARCHAR(15),
email VARCHAR(100),address VARCHAR(255), city VARCHAR(30), district
VARCHAR(30),state varchar(30), pinCode VARCHAR(6),disease
VARCHAR(50),doctorName VARCHAR(50), doctorRegistration VARCHAR(30),
illnessStage VARCHAR(15),priority VARCHAR(20),transplantingOrgan
VARCHAR(10));"""
            self.cursor.execute(query)
            self.connection.commit()
            query = """CREATE TABLE IF NOT EXISTS loginId(userName
varchar(10),password varchar(10));"""
            self.cursor.execute(query)
            query = """SELECT * FROM loginId;"""
            self.cursor.execute(query)
            if not self.cursor.fetchone():
                query = """INSERT INTO loginId(userName, password)
values('Admin','Admin@0');"""
                self.cursor.execute(query)
                self.connection.commit()
        except Exception as e:
            print(e)

def add_record(self):
        self.ui.emailVerify.setText(" ")
        if self.ui.formSelecter.currentIndex() == 1:
            data = []
            registration_number = 'REC1'
            self.cursor.execute("select registrationNumber from receiver;")
            if self.cursor.fetchone():
                self.cursor.execute("select registrationNumber from receiver ORDER BY
registrationNumber DESC LIMIT 1;")
                old_registration_number = self.cursor.fetchone()
                registration_number = int(re.search(r'\d+',
old_registration_number[0]).group())
                registration_number = 'REC' + str(registration_number + 1)
            data.append(registration_number)
            data.append(self.ui.fristName.text())
            data.append(self.ui.middleName.text())
            data.append(self.ui.lastName.text())
            date_of_birth = self.ui.dateOfBirth.date().toPyDate()
            data.append(date_of_birth.strftime("%d-%m-%Y"))
            if self.ui.radioFemale.isChecked():
                data.append('Female')
            if self.ui.radioMale.isChecked():
                data.append('Male')
            else:
                data.append('Other')
            data.append(self.ui.bloodGroup.currentText())
            data.append(self.ui.mobileNumber.text())
            data.append(self.ui.landLineNumber.text())
            data.append(self.ui.email.text())
            s = str(self.ui.email.text())
            if not s.endswith('@gmail.com'):
                self.ui.emailVerify.setText("Invalid Email id")
                return False
            data.append(self.ui.permanantAddress.toPlainText())
            data.append(self.ui.addCity.text())
            data.append(self.ui.addState.currentText())
            data.append(self.ui.addDistrict.text())
            data.append(self.ui.addPincode.text())
            data.append(self.ui.patientDisease.text())
            data.append(self.ui.doctorName.text())
            data.append(self.ui.doctorRegistration.text())
            data.append(self.ui.illnessStage.currentText())
            data.append(self.ui.patientPriority.currentText())
            data.append(self.ui.transplantOrgan.currentText())
            query = """INSERT INTO receiver(registrationNumber , firstName,
            middleName , lastName , dateOfBirth , gender , bloodGroup , mobileNumber ,
            landLineNumber , email , address , city , district , state, pinCode, disease,
            doctorName, doctorRegistration,illnessStage,priority, transplantingOrgan)
            VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);"""
             self.cursor.execute(query, data)
             self.connection.commit()
             word = data[1]
             word2 = data[2]
             word = word[2:5] + word2[2:5]
             response = add_sucess(registration_number, word)
             self.add_login_id(registration_number, word)
             if response:
                 self.renew_application()

def add_login_id(self, registration, word):
        query = """insert into loginId (userName, password) values(?,?);"""
        self.cursor.execute(query, (registration, word))
        return True
```

```
def populate_receiver(self, user_id):
        try:
           query = """ select registrationNumber, transplantingOrgan, priority,
           bloodGroup, doctorName from receiver;"""
           self.cursor.execute(query)
           waiting_table = self.cursor.fetchall()
           required_rows = len(waiting_table)
           self.ui.receiverLoginStackTable.setRowCount(required_rows)
           x = -1
           for i in waiting_table:
               x = x + 1
               for j in range(0, 5):
                     self.ui.receiverLoginStackTable.setItem(x, j,
           QtWidgets.QTableWidgetItem(i[j]))
           query = """select transplantingOrgan from receiver where registrationNumber
           =?;"""
            self.cursor.execute(query, (user_id,))
            value = self.cursor.fetchall()
            query = """select registrationNumber from receiver where transplantingOrgan
             = ? ; """
            self.cursor.execute(query, value[0])
            organ = self.cursor.fetchall()
            count = 1
            for i in organ:
                if i[0] is None:
                    continue
                elif i[0] == user_id:
                    self.ui.place.setText(str(count))
                else:
                    count = count + 1
        except Exception as e:
            print(e)


if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    win = Test()
    sys.exit(app.exec_())
```

6. SCREENSHOTS

6.1 Home Page



Organ Donation

Every day almost 18 people die waiting for organs such as heart,lungs,liver,kidney and tissues.
A single donor can save the life of 8 people and provide tissues for 7.5 others.
Organ donation is considered as charitable act and global deed.

Organ donors are needed to save thousands of lives every year .
Agreeing to donate organ does not affect the quality of care of a person receives.

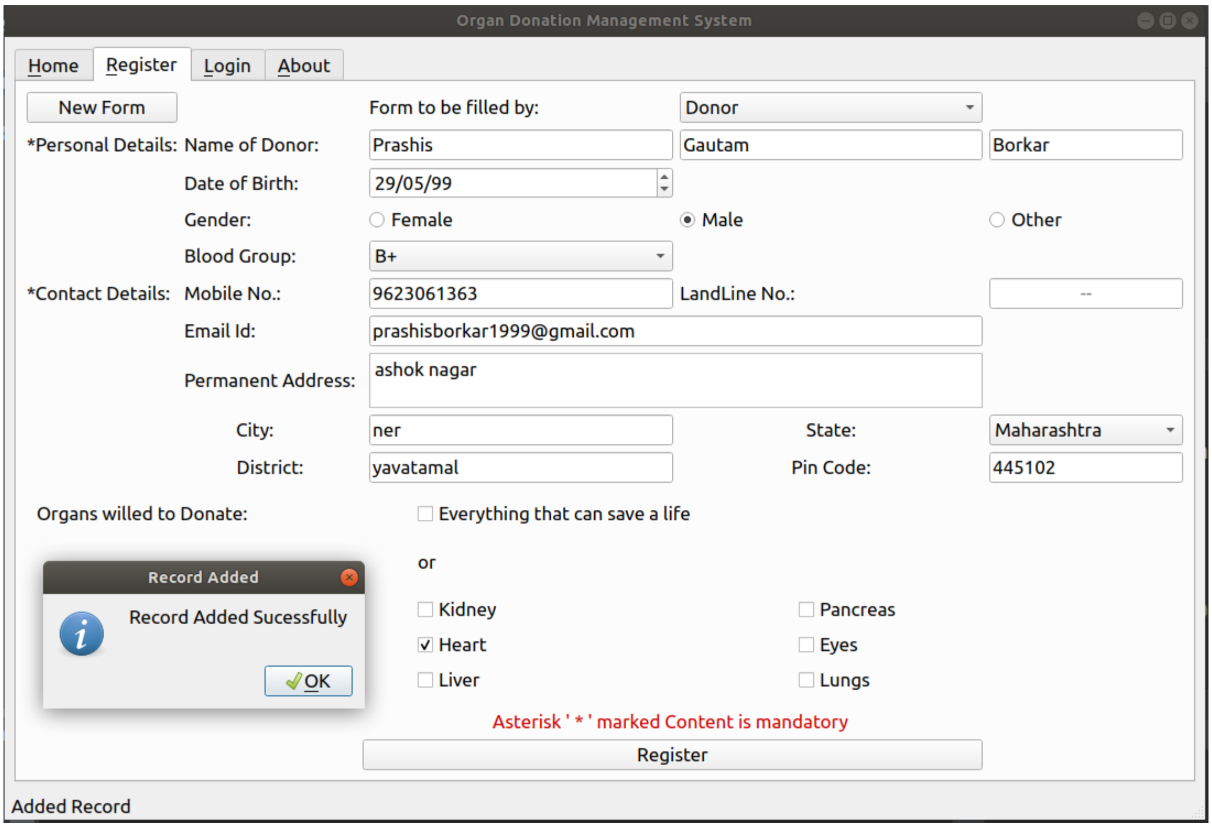
What organs can be transplanted:
♦Liver ♦Heart ♦Cornea
♦Lung ♦Intestine ♦Pancreas
♦Kidney

Who can be an organ donor?
People of all ages should consider themselves potential donor.When a person dies,he or she is evaluated for donor suitability based on their medical history and age.Anyone younger than 18 needs to have thye agreement of a parent or guardian to be a donor.Having a serious condition like actively spreading cancer,HIV,diabetes,kidney disease,or heart disease can prevent you from donating as a living donor.

In life,there are consequences...if we choose to harm ourselves because we were mindless of what could happen later,then at-least let's face the outcome....

Connected to Database

6.2 Receiver Registration



Form to be filled by: Donor

*Personal Details: Name of Donor: Prashis Gautam Borkar

Date of Birth: 29/05/99

Gender: Female [✓ Male] Other

Blood Group: B+

*Contact Details: Mobile No.: 9623061363 LandLine No.: --

Email Id: prashisborkar1999@gmail.com

Permanent Address: ashok nagar

City: ner State: Maharashtra

District: yavatamal Pin Code: 445102

Organs willed to Donate: ☐ Everything that can save a life

or

☐ Kidney ☐ Pancreas
☑ Heart ☐ Eyes
☐ Liver ☐ Lungs

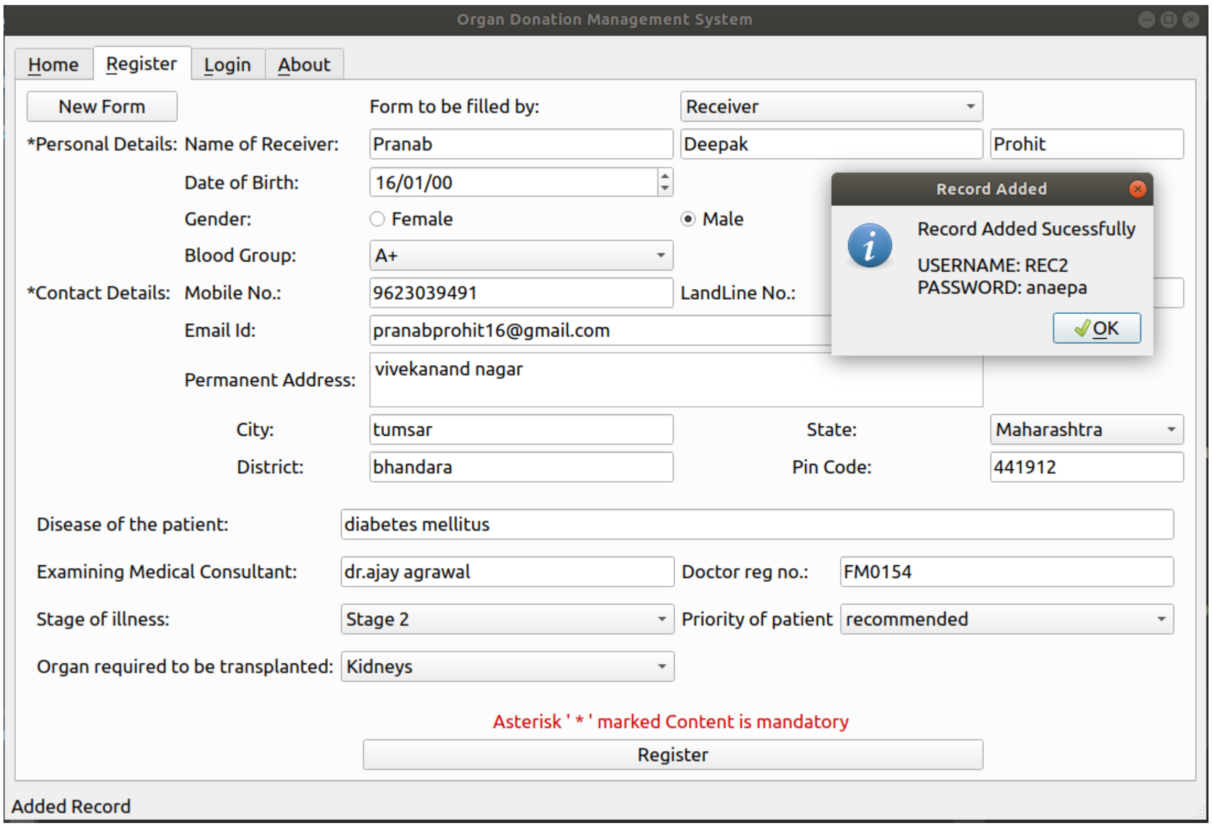
Record Added

Record Added Sucessfully

Asterisk ' * ' marked Content is mandatory

Added Record

6.3 Receiver Login



Form to be filled by: Receiver

*Personal Details: Name of Receiver: Pranab | Deepak | Prohit

Date of Birth: 16/01/00

Gender: ○ Female ● Male

Blood Group: A+

*Contact Details: Mobile No.: 9623039491 LandLine No.:

Email Id: pranabprohit16@gmail.com

Permanent Address: vivekanand nagar

City: tumsar State: Maharashtra

District: bhandara Pin Code: 441912

Disease of the patient: diabetes mellitus

Examining Medical Consultant: dr.ajay agrawal Doctor reg no.: FM0154

Stage of illness: Stage 2 Priority of patient recommended

Organ required to be transplanted: Kidneys

Asterisk ' * ' marked Content is mandatory

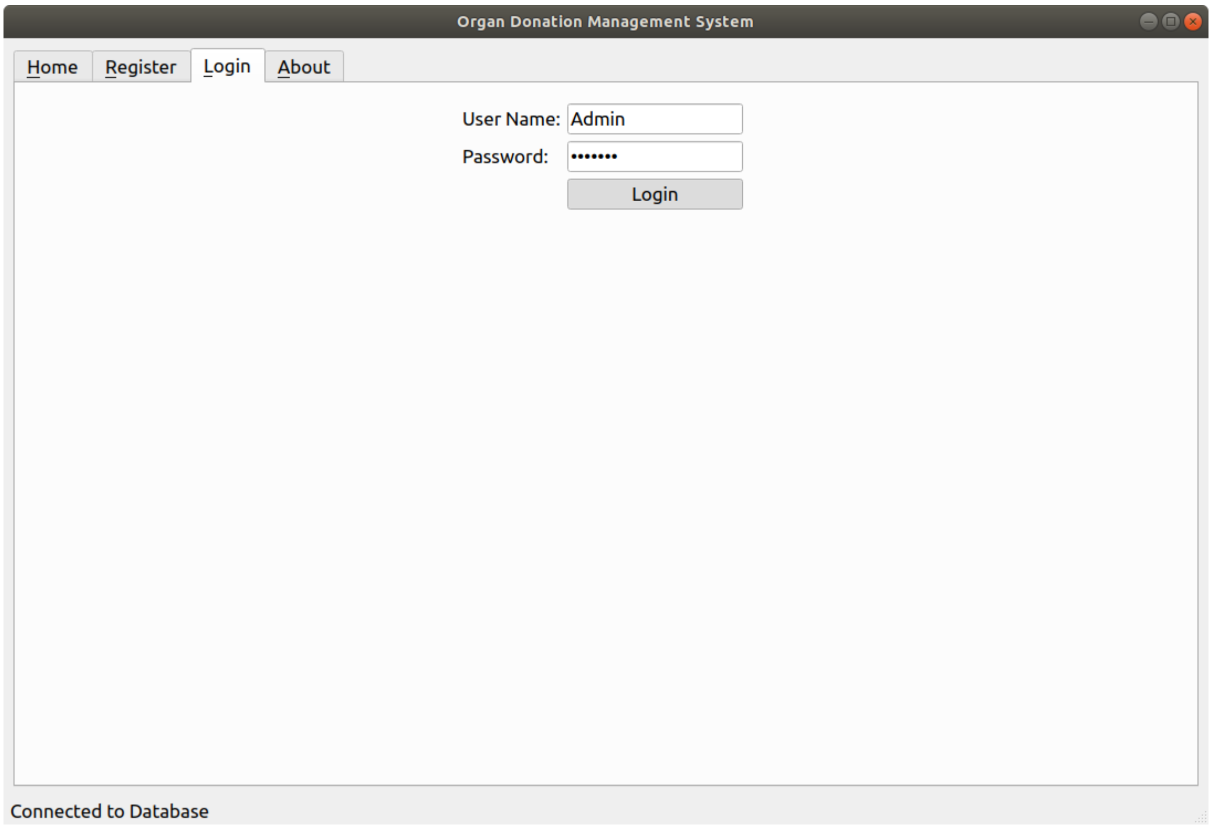
Record Added

Record Added Sucessfully

USERNAME: REC2
PASSWORD: anaepa

Added Record

6.4 Admin Login

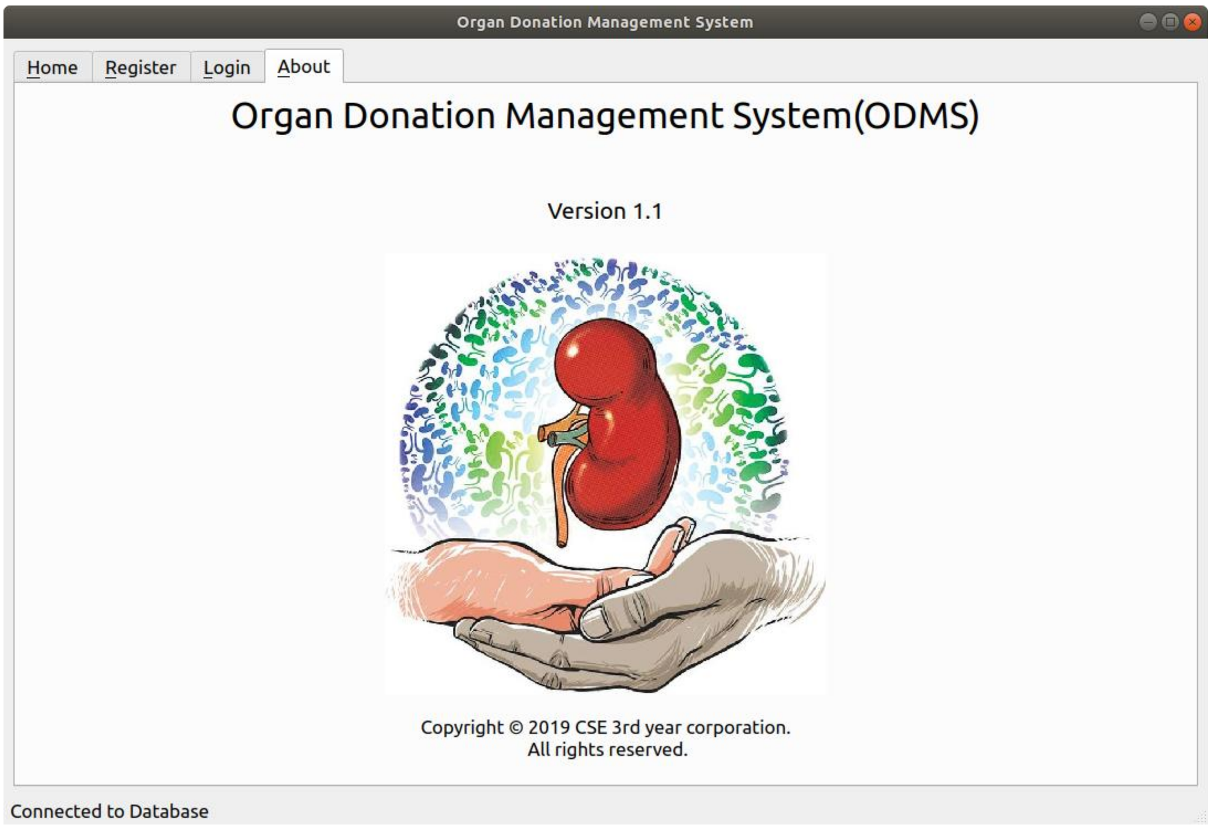


User Name: Admin

Password: •••••••

Connected to Database

6.5 About Window



Organ Donation Management System(ODMS)

Version 1.1

Copyright © 2019 CSE 3rd year corporation.
All rights reserved.

Connected to Database

7. SYSTEM REQUIREMENTS

7.1 Software Requirements

The minimum software requirement specifications for developing this project are as follows:

Programming Language:	Python 3.6
Development Environment:	PyCharm Community Edition IDE of Jet Brains
Backend Database:	SQLite 3
UI Interfacing:	Qt Designer
VCS (Version Control System):	GITHUB of Microsoft Corporation.

7.2 Hardware Requirements

The minimum hardware requirement specifications for developing this project are as follows:

Processor:	Intel core Duo 1.2 GHz or AMD Ryzen 3 2.0 GHz or greater
Hardisk:	20 MB of free space
Ram:	2 GB DDR3 or greater
Host OS:	Windows 7 SP1 or Ubuntu Linux 16.07 or greater

8. ADVANTAGES AND DISADVANTAGES

8.1 Advantages

- Efficient management of Organ Donor and Receiver Profiles
- Decrease in Data Redundancy
- Increase in data consistency
- Implementation of Government Campaigns become Flawless
- Hospitals and Organ Donation Centre can have centralized Records

8.2 Disadvantages

- Users should have some prior knowledge of handling Computers
- Authorized users should only have access to system
- Non maintenance of System can lead to data corruption or data lose.

9. CONCLUSION

Hence we have successfully developed Organ Donation Management System using the python 3.7 programming language. This software meets the objective and goals proposed earlier in the report. The user will find it useful compared to any other software in the society as it provides the better features with lesser complexities and better user interfaces.

Also during this project we have learned process of developing the software such as software documentation, team-work.

10. REFERENCES

- https://en.wikipedia.org/wiki/Organ_donation
- https://www.mohanfoundation.org/
- https://www.python.org/
- https://riverbankcomputing.com/software/pyqt/intro
- https://www.youtube.com/results?search_query=sqlite3+python+tutorial
- https://sci-hub.tw/https://ieeexplore.ieee.org/document/7930122