|  |
| --- |
| **Ambulance Service Management System** |
| DSA Final Term Project |

|  |
| --- |
| **Submitted to: Mr. Samyan Qayyum Wahla**  **Submitted** **by**: **G43**  Member01: Mubashar Ahmad 2020-CS-34  Member02: Moiz-ul-Haq 2020-CS-19  Member03: Asad Mehmood 2020-CS-10 |

Table of Contents

[1 Description: 3](#_Toc90836748)

[2 Business Case: 4](#_Toc90836749)

[3 End-User: 4](#_Toc90836750)

[4 The motivation of the Project: 4](#_Toc90836751)

[5 Level of Impact: 4](#_Toc90836752)

[6 Data Structures: 4](#_Toc90836753)

[7 Project Design: 5](#_Toc90836754)

[8 Language of Project: 5](#_Toc90836755)

[9 Work of each member: 5](#_Toc90836756)

[10 Project File Structure: 6](#_Toc90836757)

[11 Date of Final Commit: 6](#_Toc90836758)

[12 GitLab Commits: 7](#_Toc90836759)

[13 List of Features: 7](#_Toc90836760)

[14 UI: 8](#_Toc90836761)

[14.1 Wireframes: 8](#_Toc90836762)

[14.1.1 Login window: 8](#_Toc90836763)

[14.1.2 Admin Dashboard: 8](#_Toc90836764)

[14.1.3 Add Staff (Admin): 9](#_Toc90836765)

[14.1.4 View/Update Staff (Admin): 9](#_Toc90836766)

[14.1.5 Add Hospital (Admin): 10](#_Toc90836767)

[14.1.6 View/Update Hospital (Admin): 10](#_Toc90836768)

[14.1.7 View Patient Details (Admin): 11](#_Toc90836769)

[14.1.8 Add Ambulance (Admin): 11](#_Toc90836770)

[14.1.9 View/Update Ambulance (Admin): 12](#_Toc90836771)

[14.1.10 View Reports (Admin): 12](#_Toc90836772)

[14.1.11 Dashboard (Staff Member): 13](#_Toc90836773)

[14.1.12 Answer to the Call (Staff Member): 13](#_Toc90836774)

[14.1.13 Assign Ambulance (Staff Member): 14](#_Toc90836775)

[14.1.14 Call for Ambulance (Caller): 14](#_Toc90836776)

[14.1.15 View Map (Caller): 15](#_Toc90836777)

[14.2 Implementation: 15](#_Toc90836778)

[14.2.1 Start Window: 15](#_Toc90836779)

[14.2.2 Log in Window: 16](#_Toc90836780)

[14.2.3 Admin Dashboard Window: 17](#_Toc90836781)

[14.2.4 Add Staff Window (Admin): 18](#_Toc90836782)

[14.2.5 View/Update Staff (Admin): 19](#_Toc90836783)

[14.2.6 Add Hospital Window (Admin): 20](#_Toc90836784)

[14.2.7 View/Update Hospital Window (Admin): 21](#_Toc90836785)

[14.2.8 View Patient Details Window (Admin): 22](#_Toc90836786)

[14.2.9 Add Ambulance Window (Admin): 23](#_Toc90836787)

[14.2.10 View/Update Ambulance (Admin): 24](#_Toc90836788)

[14.2.11 View Reports Window (Admin): 25](#_Toc90836789)

[14.2.12 Staff Member Dashboard Window: 26](#_Toc90836790)

[14.2.13 Answer to the Call Window (Call Receiver): 27](#_Toc90836791)

[14.2.14 Assign Ambulance Window (Call Receiver): 28](#_Toc90836792)

[14.2.15 Call for Ambulance Window (Caller): 29](#_Toc90836793)

[14.2.16 View Map Window (Admin): 30](#_Toc90836794)

[15 Complete Working: 31](#_Toc90836795)

[16 Flow Diagram: 34](#_Toc90836796)

[17 Class Diagrams: 35](#_Toc90836797)

# Table of Figures:

[Figure 1: Project File Structure 6](#_Toc90836814)

[Figure 2: Login Window (Wireframe) 8](#_Toc90836815)

[Figure 3: Admin Dashboard (Wireframe) 8](#_Toc90836816)

[Figure 4: Add Staff (Admin) (Wireframe) 9](#_Toc90836817)

[Figure 5:View/Update Staff (admin) (Wireframe) 9](#_Toc90836818)

[Figure 6: Add Hospital (Admin) (Wireframe) 10](#_Toc90836819)

[Figure 7: View/Update Hospital(Admin) (Wireframe) 10](#_Toc90836820)

[Figure 8: View Patient Details (Wireframe) 11](#_Toc90836821)

[Figure 9: Add Ambulance (Wireframe) 11](#_Toc90836822)

[Figure 10: View / Update Ambulance (Wireframe) 12](#_Toc90836823)

[Figure 11: View Reports (Wireframe) 12](#_Toc90836824)

[Figure 12: Dashboard (Wireframe) 13](#_Toc90836825)

[Figure 13: Answer to the call (Wireframe) 13](#_Toc90836826)

[Figure 14:Assign Ambulance (Wireframe) 14](#_Toc90836827)

[Figure 15: Call For Ambulance (Wireframe) 14](#_Toc90836828)

[Figure 16: View Map (Wireframe) 15](#_Toc90836829)

[Figure 17: Start Window (GUI) 15](#_Toc90836830)

[Figure 18: Login Window (GUI) 16](#_Toc90836831)

[Figure 19: Dashboard Window (GUI) 17](#_Toc90836832)

[Figure 20: Add Staff Window (GUI) 19](#_Toc90836833)

[Figure 21: View/Update Staff Window (GUI) 20](#_Toc90836834)

[Figure 22: Add Hospital Window (GUI) 21](#_Toc90836835)

[Figure 23: View/Update Hospital Window (GUI) 22](#_Toc90836836)

[Figure 24: View Patients Details Window (GUI) 23](#_Toc90836837)

[Figure 25: Add Ambulance Window (GUI) 24](#_Toc90836838)

[Figure 26: View Update Ambulance Window (GUI) 25](#_Toc90836839)

[Figure 27: View Reports Window (GUI) 26](#_Toc90836840)

[Figure 28: Staff Member Dashboard (GUI) 27](#_Toc90836841)

[Figure 29: Answer to the Call (GUI) 28](#_Toc90836842)

[Figure 30: Assign Ambulance (GUI) 29](#_Toc90836843)

[Figure 31: Call for Ambulance (GUI) 30](#_Toc90836844)

[Figure 32: View Map (GUI) 31](#_Toc90836845)

[Figure 33: Flow diagram of complete project 34](#_Toc90836846)

# Description:

This project is aimed to develop such a system that manages the ambulance services provided by the healthcare organization. This project will make sure the availability of ambulance services. This system will provide emergency medical services like first aid, ambulances, etc. This system has three types of users Admin, Staff Member, User (Person who calls for an ambulance). Admin can add, view, and update staff member which has different categories like a driver, doctor, caller receiver, etc. Also, the admin can add, view, and update hospitals in any city or specific district. Admin can also view patient details which contain a number of patients brought by ambulances in any specific hospital. Admin can also add, view, and update ambulance. This system will also keep track of the ambulances. When a caller will make a call to call an ambulance, he will be redirected to the page where he will be asked to enter his city name, then in the nearest control room of the Ambulance service his/her details will be entered, and call receiver at the control room will assign an ambulance to the caller. In case, if an Ambulance is not present at that control center, then that control room will ask for an ambulance to the nearest other control room in that area. After the ambulance will assign, it will start traveling to the caller and will take the patient and will admit them to the nearest hospital. The caller and receiver both will be able to visualize the track of the ambulance. A simple map of cities and ambulances will be shown along their details.

# Business Case:

We are designing this system that will help a citizen who can face any emergency in daily life efficiently. This system will help patients to be treated on the information given at the time of emergency call quickly.

# End-User:

General Public will be the end-user.

# The motivation of the Project:

To help the general public through ambulance services and make the ambulance service system more efficient for a healthcare organization.

# Level of Impact:

This system will help rescue 1122 to recognize staff member (call receiver) performance in case of emergency of a citizen by checking response time of call receiver etc.

# Data Structures:

|  |  |
| --- | --- |
| **Data Structure** | **Description** |
| Linked list | To store Call Receiver Id’s, Drivers Id’s and Ambulances Id’s int CallCenter class.  To store callers data in CallerManager class.  To store City Vertices data in Graph implementation. Also to store CityVertices that are found in shortest path.  To store patients data in PatientManager class. |
| Dictionary | To store cities data in CityManager class, ambulances data in AmbualanceManager class, call centres data in CallCenterManager class, call receivers data in CallReceiverManager class, city vertices data in CityVertexManager class, drivers data in DriverManager class, to create adjLists of city edges in Graph, to store picture boxes data to view on the map in ViewMap form. |
| Stack | While assigning ambulances to callers in the assignAmbulance class. |
| Graph | To find the shortest path between cities. Dijkstra’s algorithm used to find out the shortes path. |
| Array | While working with file handling arrays are used to store data after reading from file. |

# Project Design:

OOP based

# Language of Project:

C#

# Work of each member:

|  |  |  |
| --- | --- | --- |
| Name | Reg. No. | Tasks Done |
| Mubashar Ahmad | 2020-CS-34 | Admin GUI, Backend for admin options(Dashboard, Add Staff, View/Update Staff, View Patients), Backend for Caller( Make a call for ambulance),Backend for Staff (Assign ambulance to the caller, dashboard), Implementation of graph(Dijkstra Shortest path algorithm), Map Visualization. |
| Moiz-ul-Haq | 2020-CS-19 | Staff GUI, Backend for admin options (Add Hospital, View/Update Hospital, View Reports), Backend for Caller (Patients Record), Backend for Staff(Answer Call), Implementation of Graph(CityVertex.cs, CityVertex Manager) |
| Asad Mehmood | 2020-CS-10 | Caller GUI, Backend for admin options(Add Ambulance, View Update Ambulances data), created CityEdges Files for graph, Implementation of Graph(Graph.cs, CityEdges.cs) |

# Project File Structure:

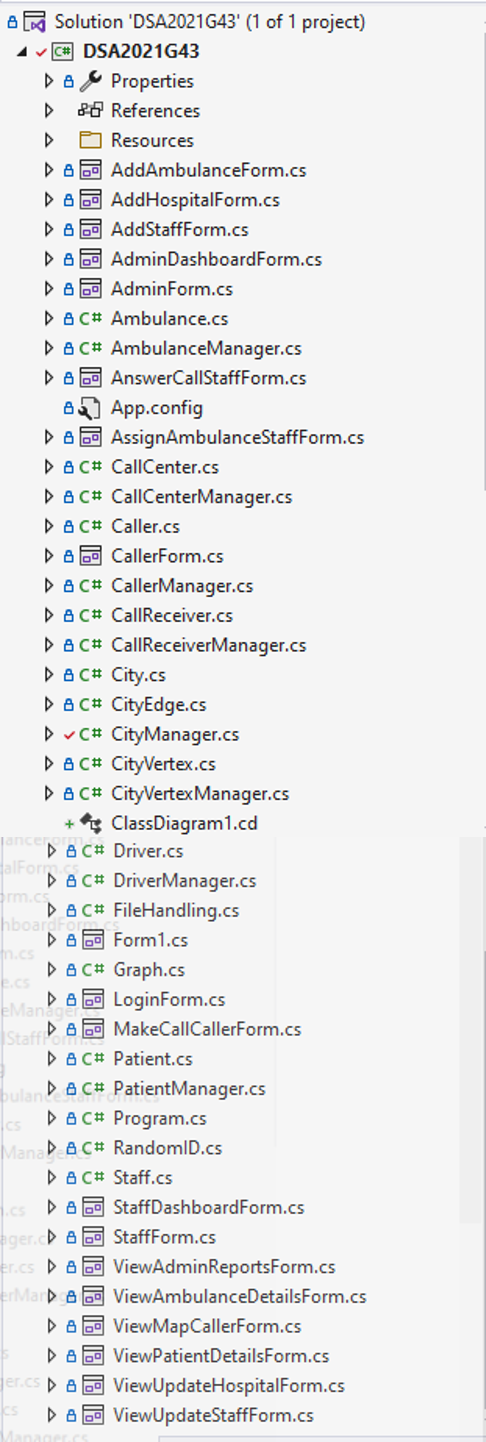


Figure 1: Project File Structure

# Date of Final Commit:

Friday 18-Dec-2021

# GitLab Commits:

|  |  |  |
| --- | --- | --- |
| Name | Reg. No. | No. of Commits |
| Mubashar Ahmad | 2020-CS-34 | 10 |
| Moiz-ul-Haq | 2020-CS-19 | 8 |
| Asad Mehmood | 2020-CS-10 | 8 |

# List of Features:

|  |  |
| --- | --- |
| Actor | Features |
| Admin | Dashboard |
| Add Staff |
| View/Update Staff |
| Add Hospital |
| View/Update Hospital |
| View Patients |
| Add Ambulance |
| View Ambulance Data |
| Generate Reports |
| Staff Member | Dashboard |
| Receive Call |
| Assign Ambulance |
| Caller | Make a Call |
| View Map |

# UI:

## Wireframes:

### Login window:

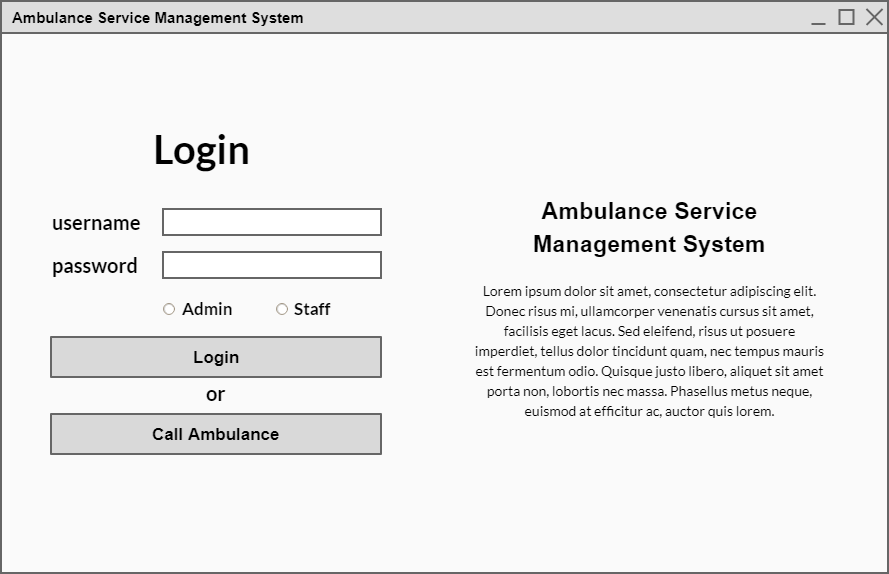


Figure 2: Login Window (Wireframe)

### Admin Dashboard:

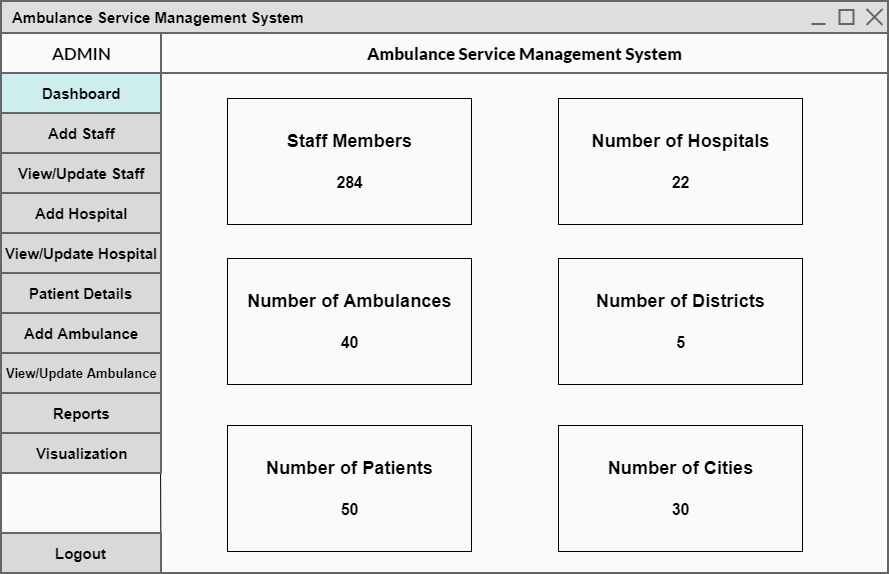


Figure 3: Admin Dashboard (Wireframe)

### Add Staff (Admin):

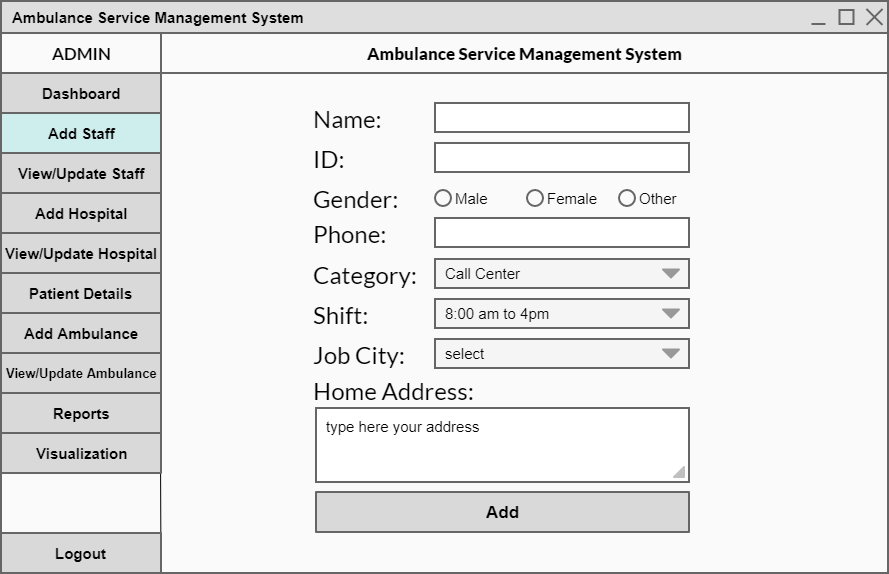


Figure 4: Add Staff (Admin) (Wireframe)

### View/Update Staff (Admin):

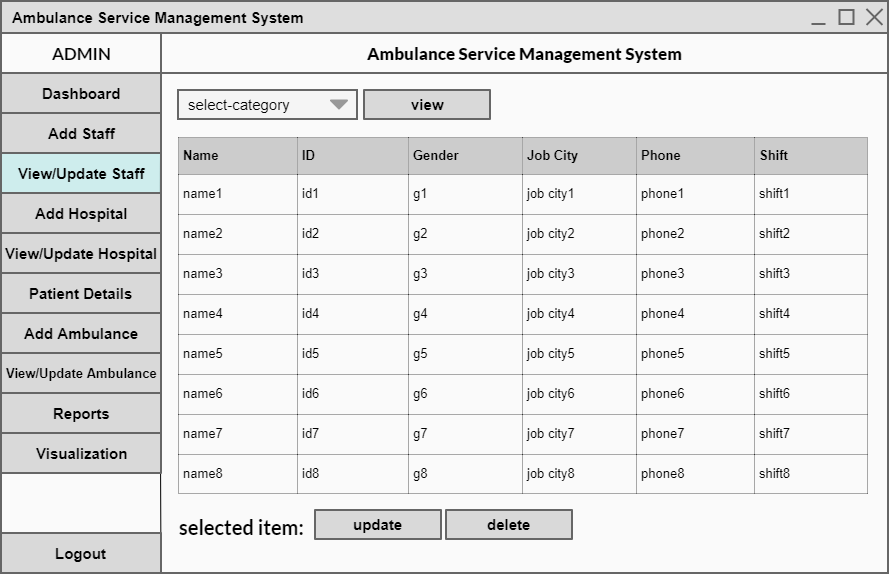


Figure 5:View/Update Staff (admin) (Wireframe)

### Add Hospital (Admin):

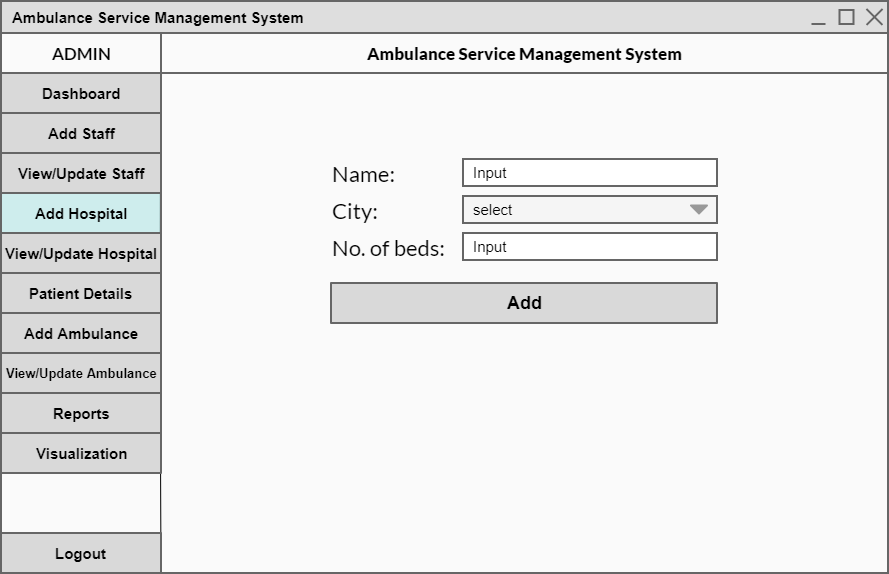


Figure 6: Add Hospital (Admin) (Wireframe)

### View/Update Hospital (Admin):

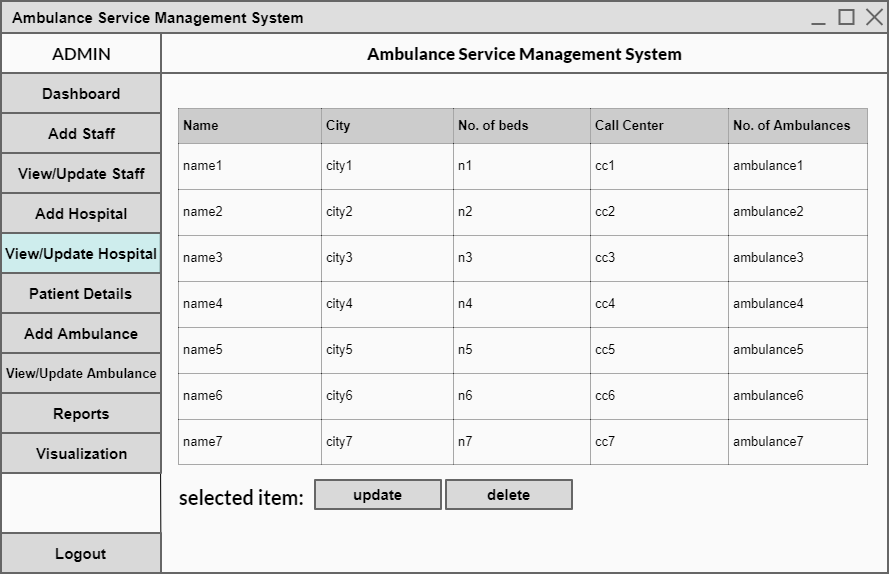
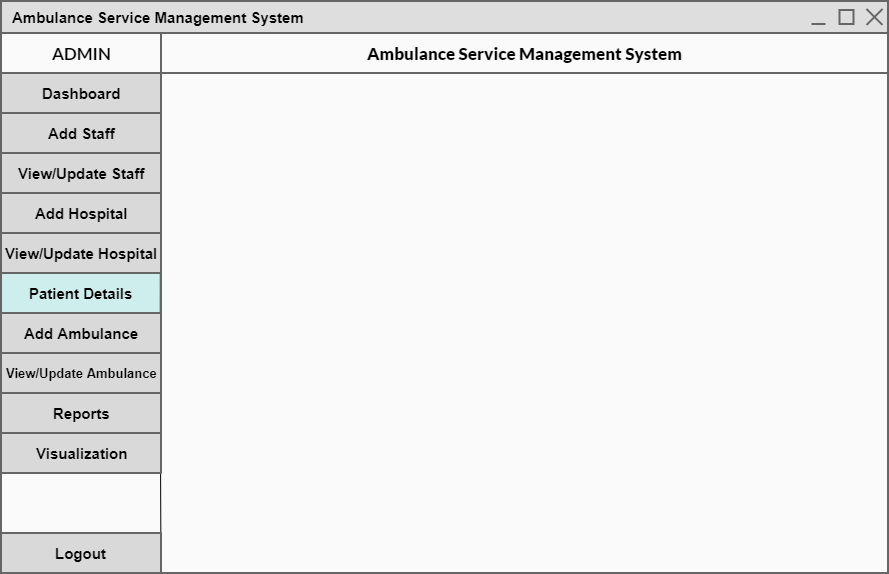


Figure 7: View/Update Hospital(Admin) (Wireframe)

### View Patient Details (Admin):



Here table will be show to view the patents

Figure 8: View Patient Details (Wireframe)

### Add Ambulance (Admin):

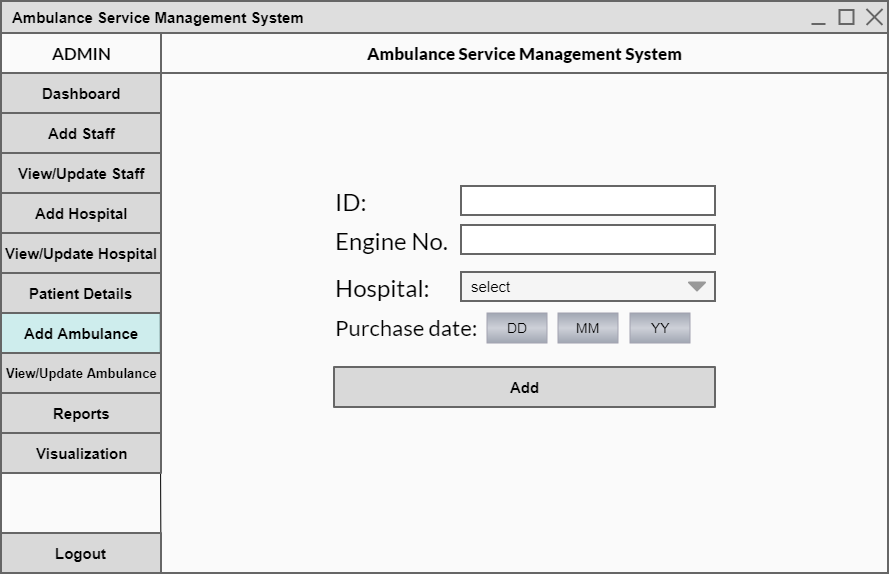


Figure 9: Add Ambulance (Wireframe)

### View/Update Ambulance (Admin):

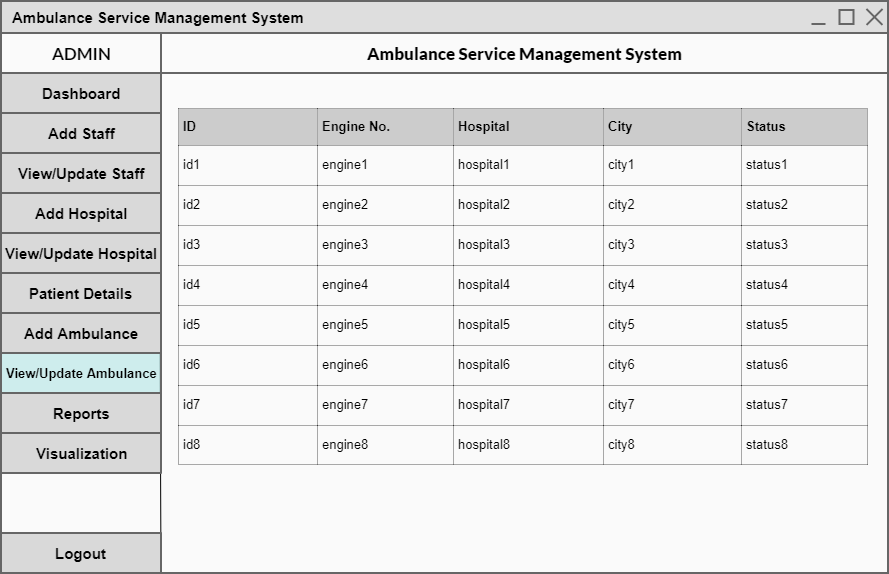


Figure 10: View / Update Ambulance (Wireframe)

### View Reports (Admin):

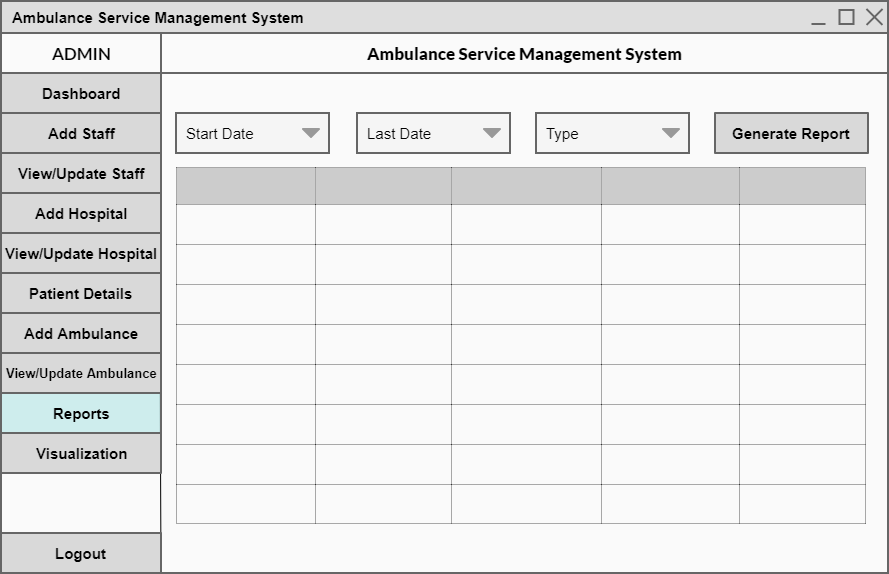


Figure 11: View Reports (Wireframe)

### Dashboard (Staff Member):

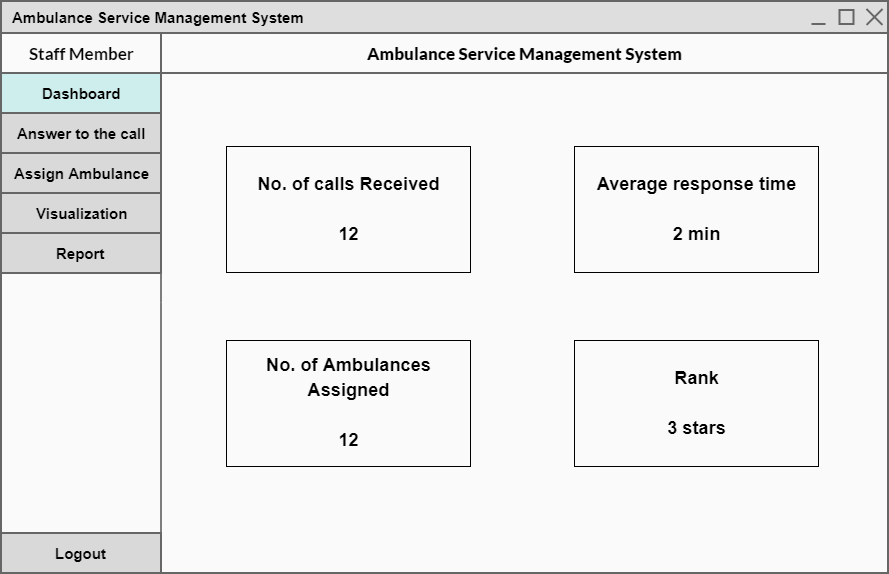


Figure 12: Dashboard (Wireframe)

### Answer to the Call (Staff Member):

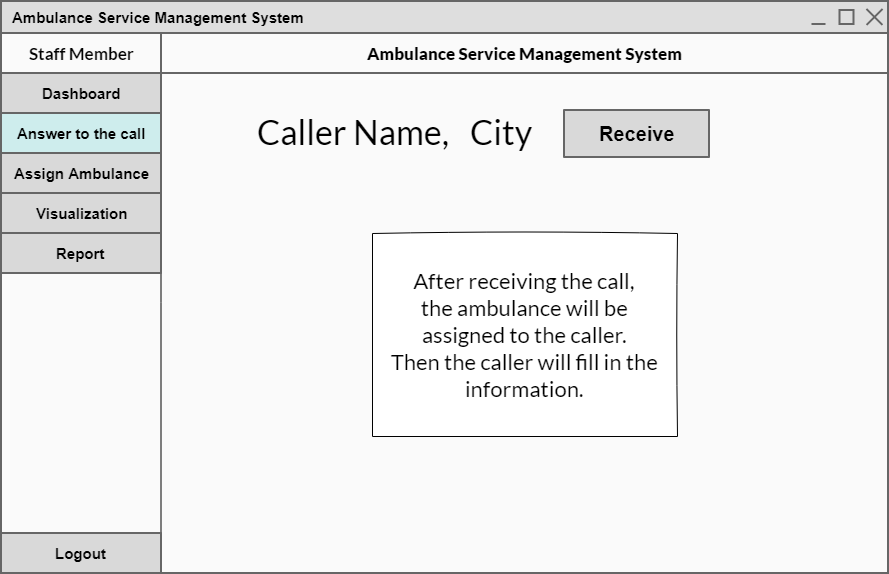


Figure 13: Answer to the call (Wireframe)

### Assign Ambulance (Staff Member):

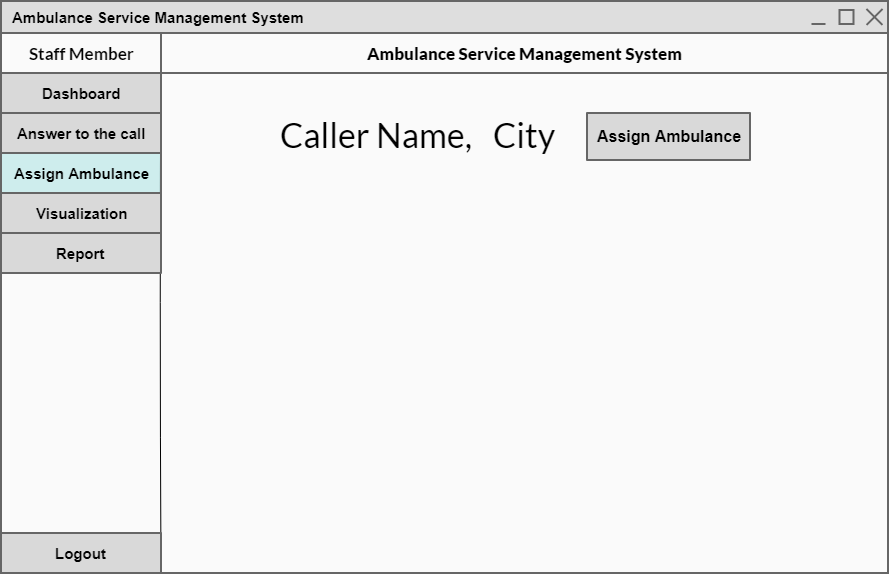


Figure 14:Assign Ambulance (Wireframe)

### Call for Ambulance (Caller):

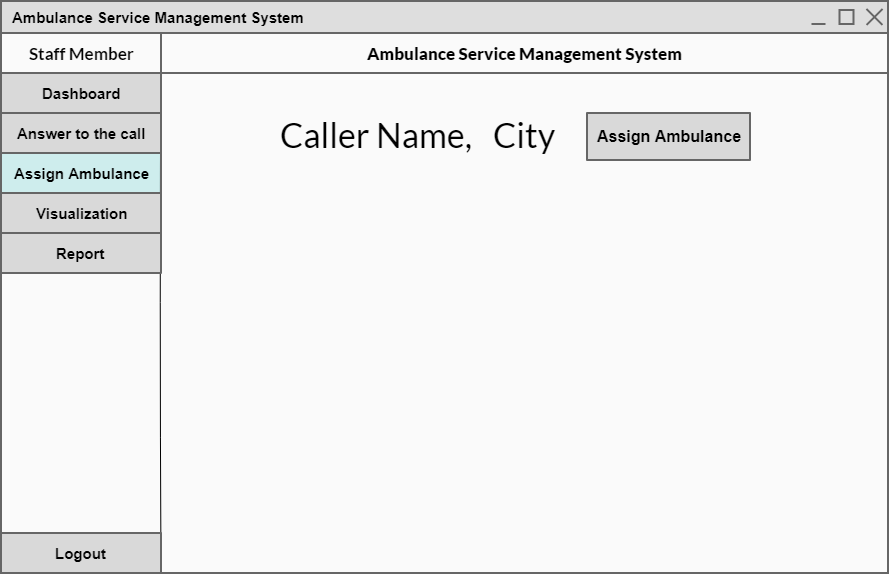


Figure 15: Call For Ambulance (Wireframe)

### View Map (Caller):

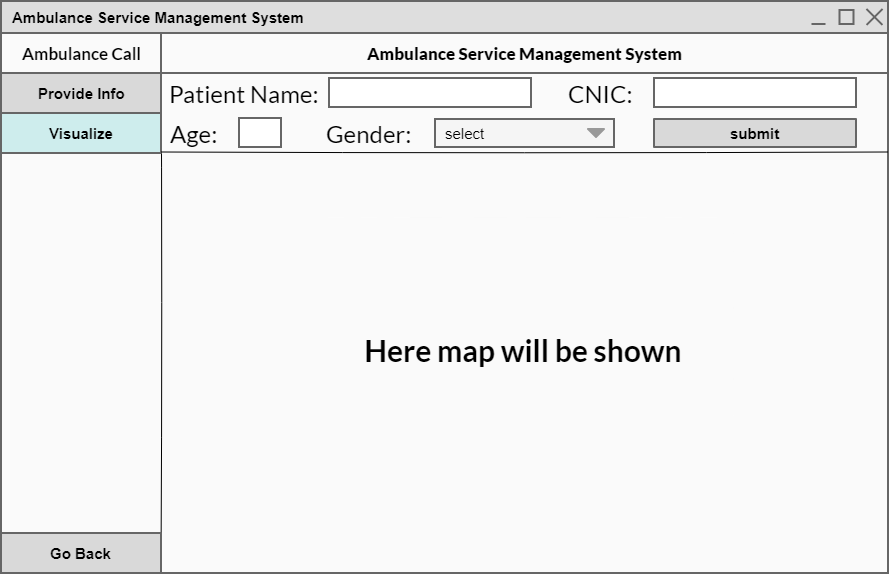


Figure 16: View Map (Wireframe)

## Implementation:

### Start Window:

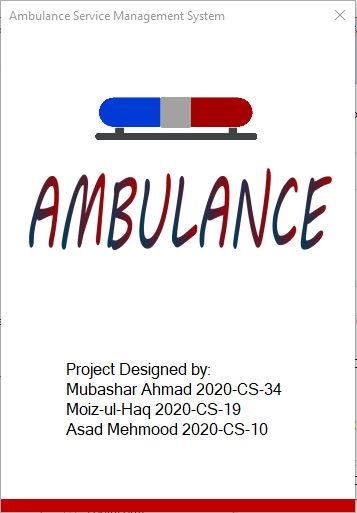


Figure 17: Start Window (GUI)

### Log in Window:

|  |  |
| --- | --- |
| Use case name | Login page |
| Actor | Admin, Staff Member, Caller |
| Flow of events | ( Username, password, Admin/staff, login ) / call ambulance |

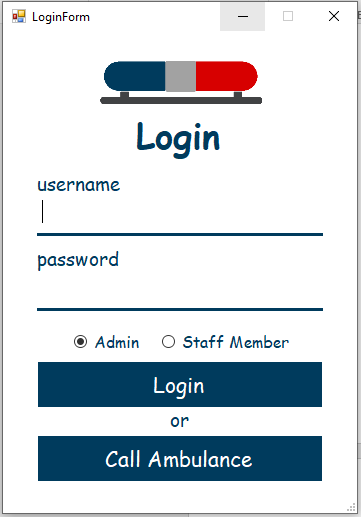


Figure 18: Login Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| login\_label | Label | Label to show text (login). |
| username\_input | TextBox | To take input username as text. |
| password\_input | TextBox | To take input password as text. |
| admin\_rb | RadioButton | To take input either he/she is admin or staff. |
| staff\_rb |
| login\_btn | Button | To login. |
| call\_ambulance\_btn | Button | To go to Caller window |

### Admin Dashboard Window:

|  |  |
| --- | --- |
| Use case name | Admin dashboard |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on the dashboard button |

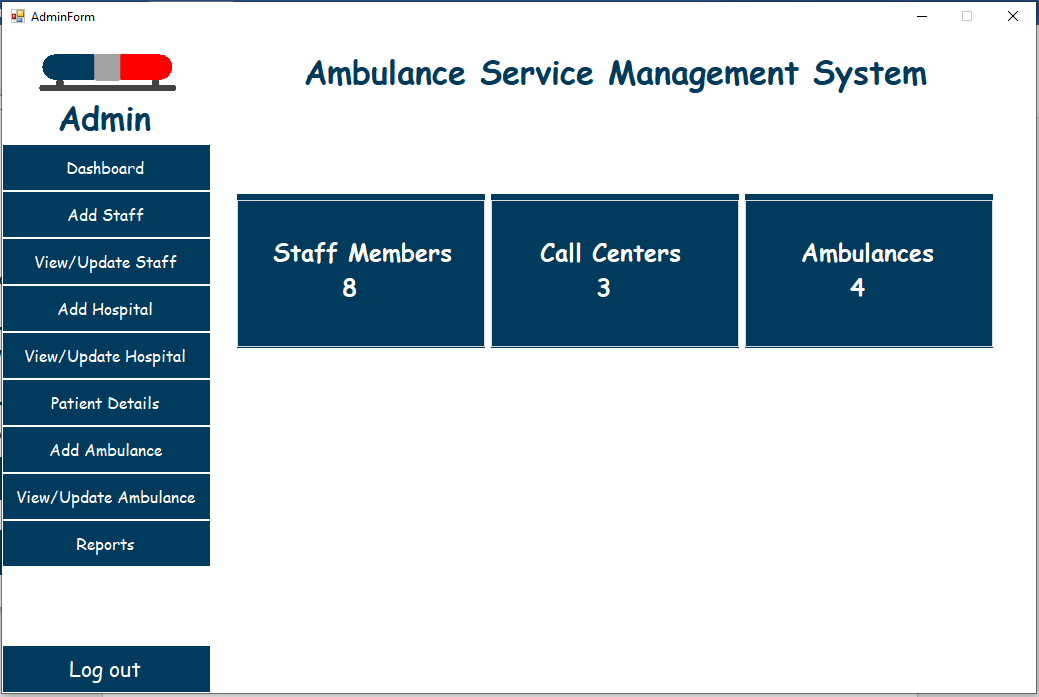


Figure 19: Dashboard Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| admin\_label | Label | To show its admin menu. |
| project\_name\_label | To show project name. |
| dashboard\_btn | Button | To show staff members, number of hospitals, ambulances, districts, patients, cities etc. |
| add\_staff\_btn | To add new staff member. |
| view/update\_staff\_btn | To view staff member and update any detail. |
| add\_hospital\_btn | To add hospital in city where hospital is not present. |
| view/update\_hospital\_btn | To view and update hospital details. |
| patients\_details\_btn | To show info related to patient. |
| add\_ambulance\_btn | To add ambulances in parking area. |
| view/update\_ambulance\_btn | To view and update info related to ambulances. |
| reports\_btn | To show reports that belongs patients, ambulances , staff, hospitals etc. |
| logout\_btn | To logout admin from his/her account. |
| staff\_members\_label | Label | To show number of staff members, ambulances and call centers |
| number\_of\_ambulances\_label |
| call\_centers\_label |

### Add Staff Window (Admin):

|  |  |
| --- | --- |
| Use case name | Add staff page |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on Add Staff button 🡪 enter name of staff member 🡪 enter id 🡪 select gender 🡪 enter phone number 🡪 select category 🡪 select shift 🡪 select job city 🡪 provide home address 🡪 click on add button |

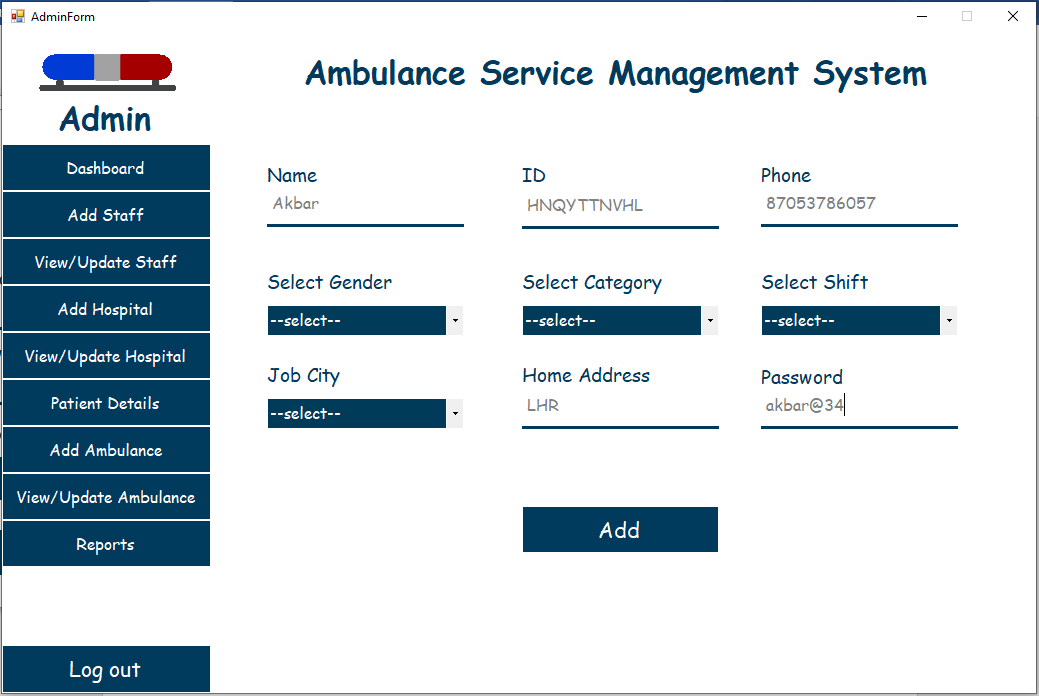


Figure 20: Add Staff Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| add\_btn | Button | To add new staff member. |
| name\_textbox | TextBox | To enter name of his/her. |
| id\_textbox | To enter ID of his/her. |
| phone\_textbox | To enter phone number. |
| home\_address\_textbox | To enter home address of staff member. |
| password\_input | To input password |
| gender\_cb | ComboBox | To take gender input. |
| shift\_combo\_box | To select shift for staff member in which he/she will perform his/her duty. |
| category\_combo\_box | To select category for staff member. |
| job\_city\_combo\_box | To select city where staff member will be appointed. |

### View/Update Staff (Admin):

|  |  |
| --- | --- |
| Use case name | View/Update staff |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on view/update staff button 🡪 select category 🡪 click on view button 🡪 select any row 🡪 click on update/delete button |

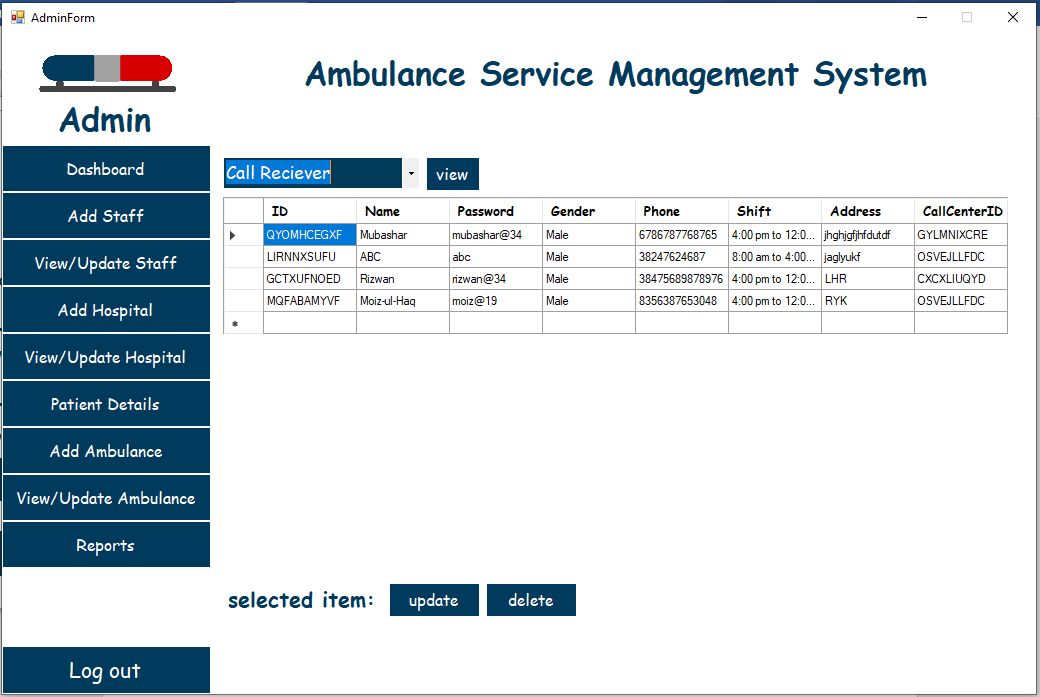


Figure 21: View/Update Staff Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| view\_btn | Button | To view/update and delete any staff member. |
| update\_btn |
| delete\_btn |
| select\_category\_combo\_box | ComboBox | To select any specific category to show data of it. |
| dataGridViewTable | DataGridView | To display data of staff members. |

### Add Hospital Window (Admin):

|  |  |
| --- | --- |
| Use case name | Add Hospital |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on Add Hospital button 🡪 enter the name of hospital 🡪 select city 🡪 enter number of beds 🡪 click on Add button |

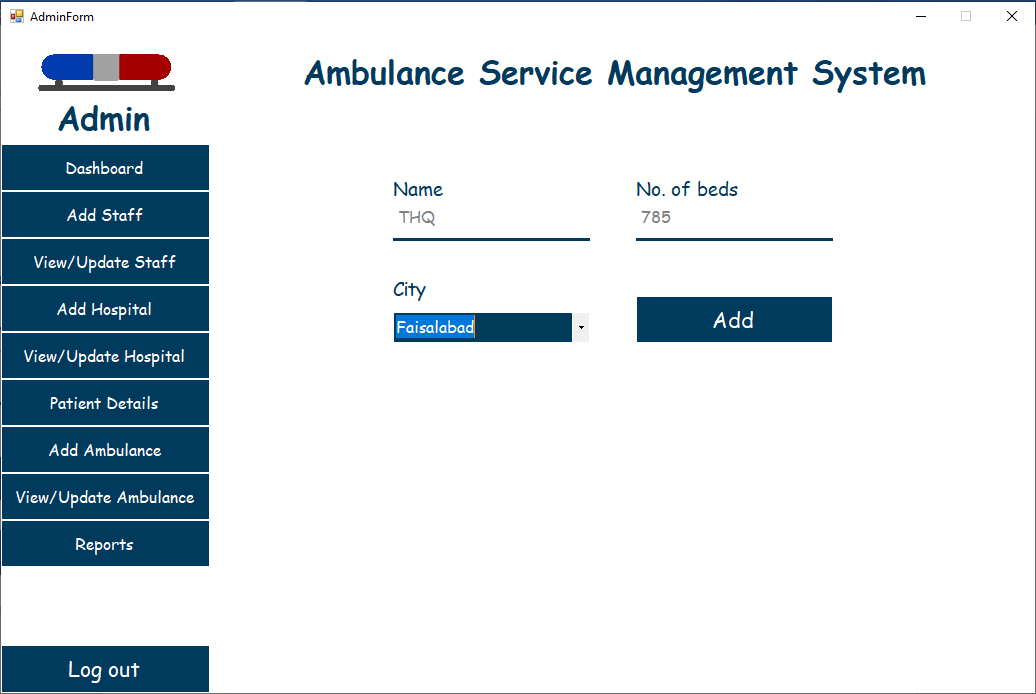


Figure 22: Add Hospital Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| add\_btn | Button | To add new hospital in any specific city. |
| name\_textbox | TextBox | To enter name of hospital. |
| number\_of\_beds\_textbox | To enter number of beds that hospital will contain. |
| city\_combo\_box | ComboBox | To select city where we want to add hospital/call center. |

### View/Update Hospital Window (Admin):

|  |  |
| --- | --- |
| Use case name | View/Update Hospital |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on view/update hospital button 🡪 select any row 🡪 click on update/delete button |

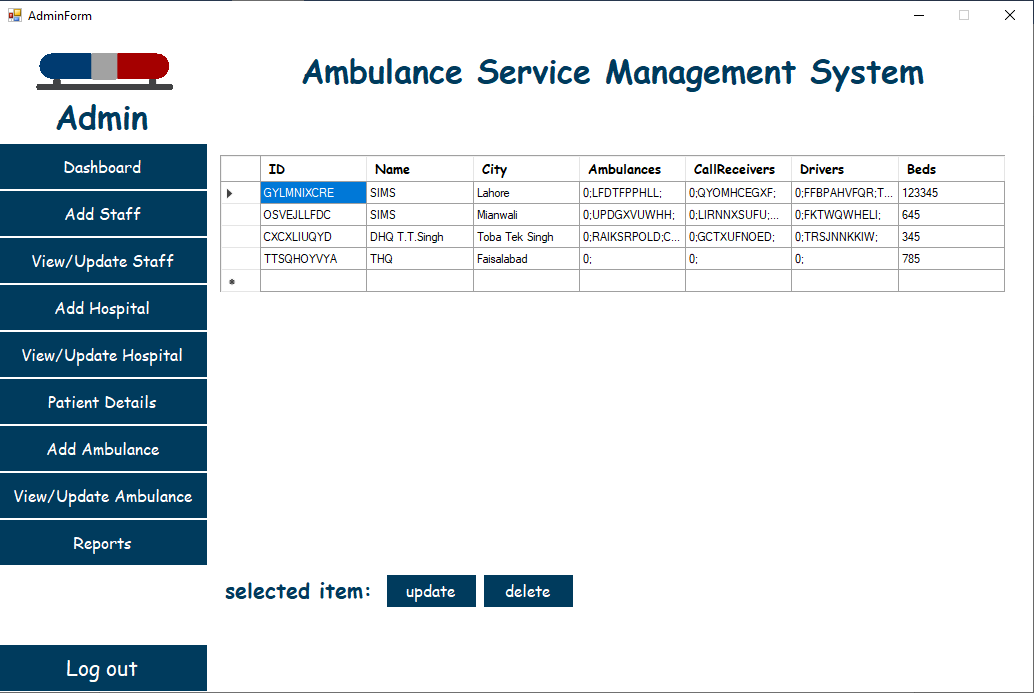


Figure 23: View/Update Hospital Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| update\_btn | Button | To update or delete any hospital details. |
| delete\_btn |
| dataGridViewTable | DataGridView | To show data of hospitals. |

### View Patient Details Window (Admin):

|  |  |
| --- | --- |
| Use case name | View patient details |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on patient details |

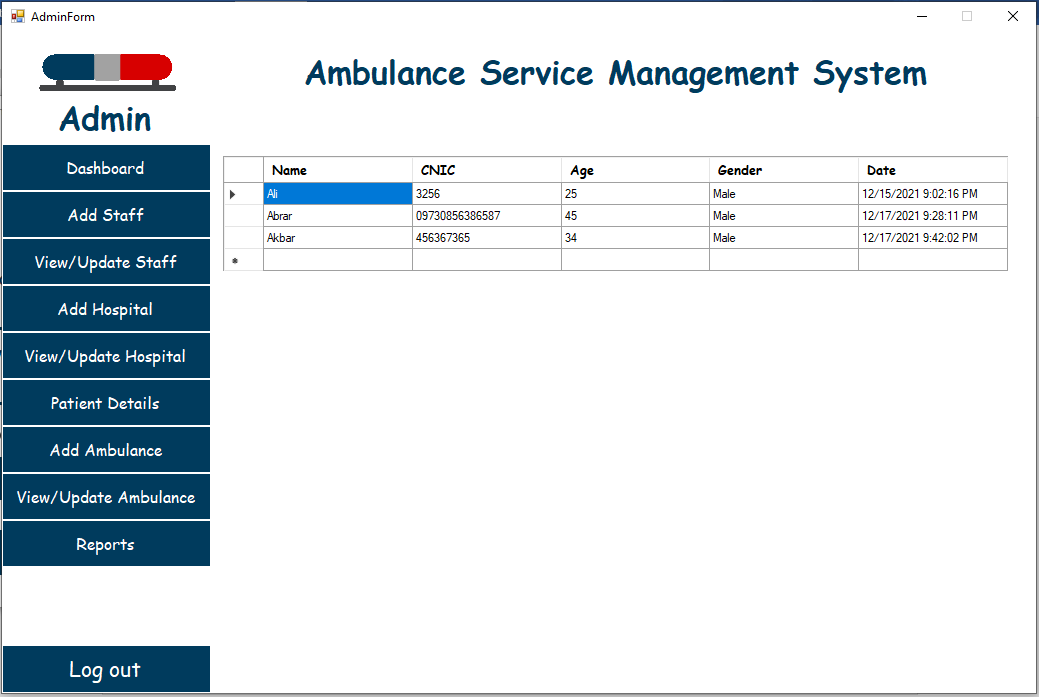


Figure 24: View Patients Details Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| dataGridViewtable | DataGridView | To show data of patients |

### Add Ambulance Window (Admin):

|  |  |
| --- | --- |
| Use case name | Add ambulance |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on add ambulance button 🡪 enter ID 🡪 Enter engine number 🡪 select hospital 🡪 provide purchase date 🡪 click on Add button |

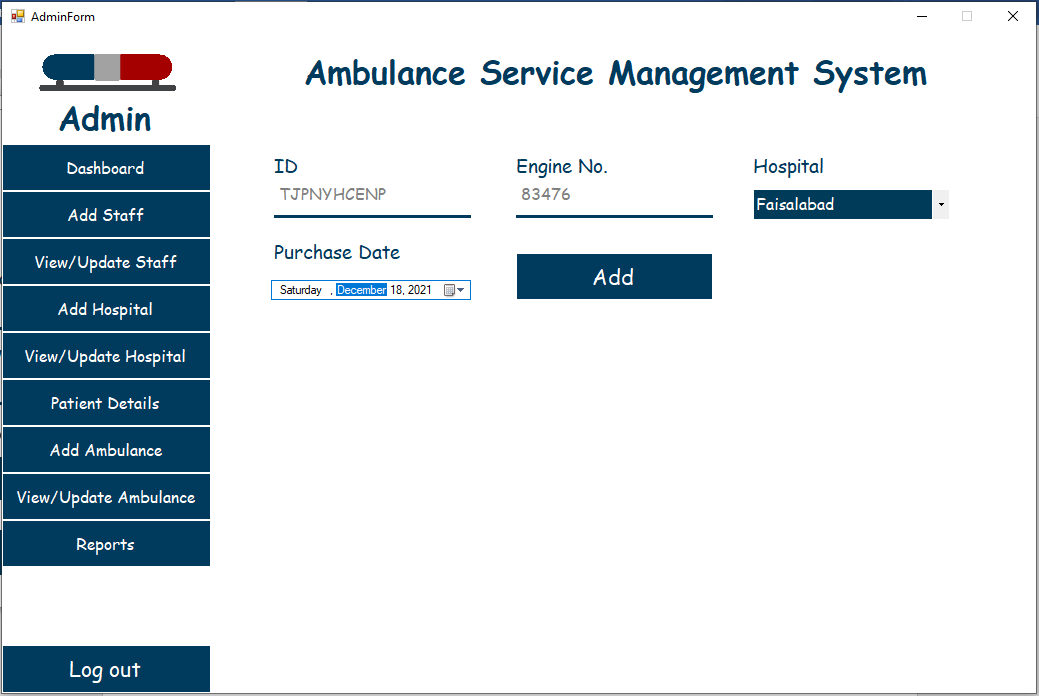


Figure 25: Add Ambulance Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| add\_btn | Button | To add new ambulance to any specific hospital. |
| id\_textbox | TextBox | To show ambulance ID. |
| engine\_no\_textbox | To enter ambulance engine number. |
| hospital\_combo\_box | ComboBox | To select any specific hospital in which ambulance will be added. |
| date\_input\_label | Label | To show text. |
| date\_input\_picker | DateTimePicker | To select date of purchase of ambulance. |

### View/Update Ambulance (Admin):

|  |  |
| --- | --- |
| Use case name | View/Update Ambulance |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on view/update button |

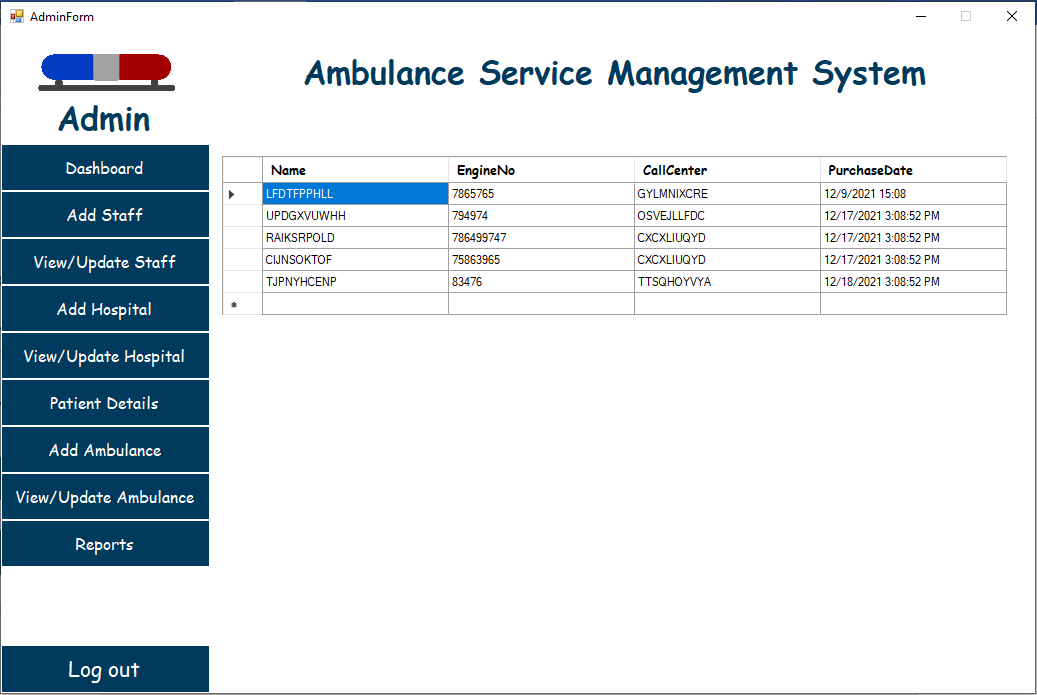


Figure 26: View Update Ambulance Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| dataGridViewTable | DataGridView | To show data that belongs to ambulances. |

### View Reports Window (Admin):

|  |  |
| --- | --- |
| Use case name | View reports |
| Actor | Admin |
| Flow of events | Login as admin 🡪 click on reports button 🡪 select start date 🡪 select end date 🡪 select category/type 🡪 click Generate Report button |

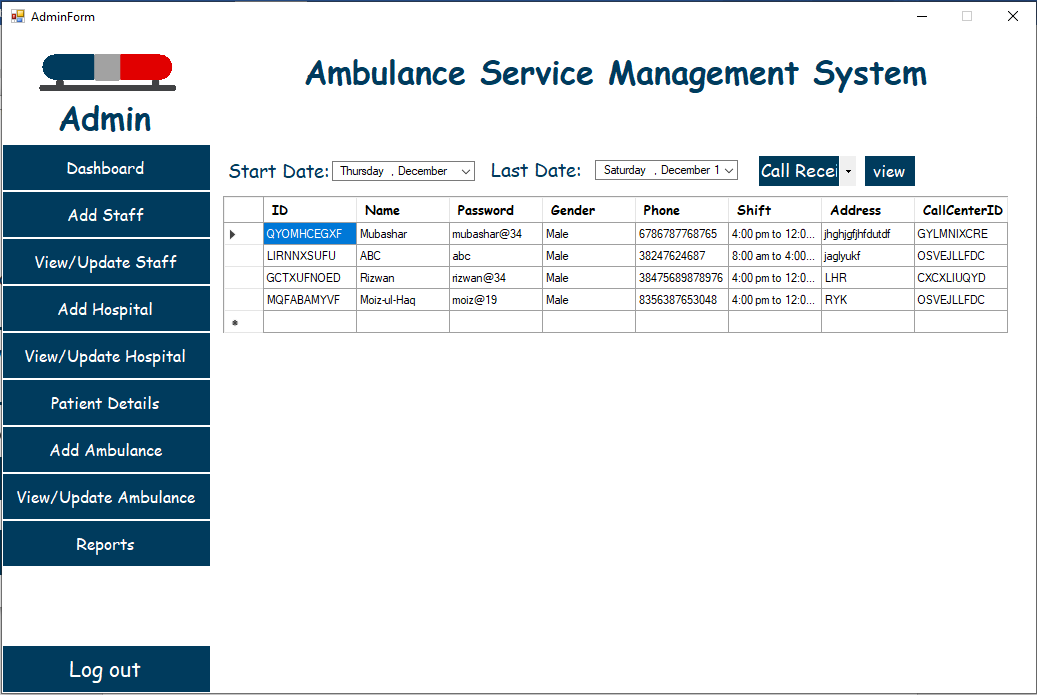


Figure 27: View Reports Window (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| dataGridViewTable | DataGridView | To show reports of patients, ambulances, hospitals, staff etc. Within specific dates. |
| view\_report\_btn | Button | To show data according to dates selected above. |
| start\_date\_combo\_box | ComboBox | To select starting/ending date and type. |
| last\_date\_combo\_box |
| type\_combo\_box |

### Staff Member Dashboard Window:

|  |  |
| --- | --- |
| Use case name | Dashboard |
| Actor | Staff Member |
| Flow of events | Log in as a staff member 🡪 click on the Dashboard button |

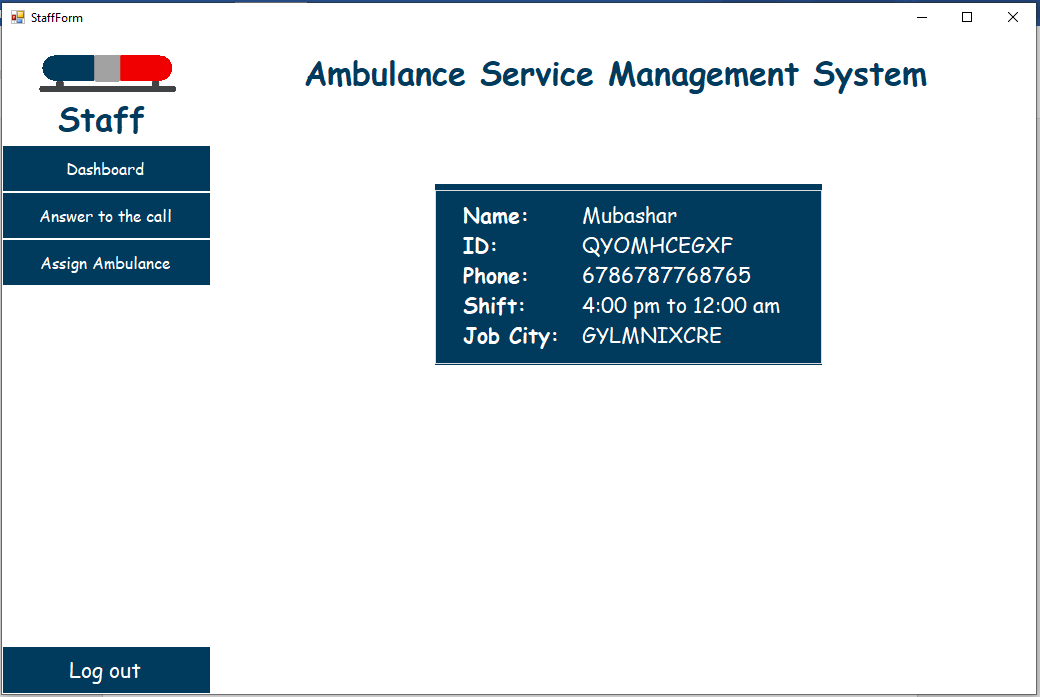


Figure 28: Staff Member Dashboard (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| name\_label | Label | To show name, id, phone, shift, and job city of the staff member that is currently logged in. |
| id\_label |
| phone\_label |
| shift\_label |
| job\_city\_label |

### Answer to the Call Window (Call Receiver):

|  |  |
| --- | --- |
| Use case name | Answer to the call |
| Actor | Staff Member |
| Flow of events | Log in as a staff member 🡪 click on the Answer to the call button 🡪 click on receive button |

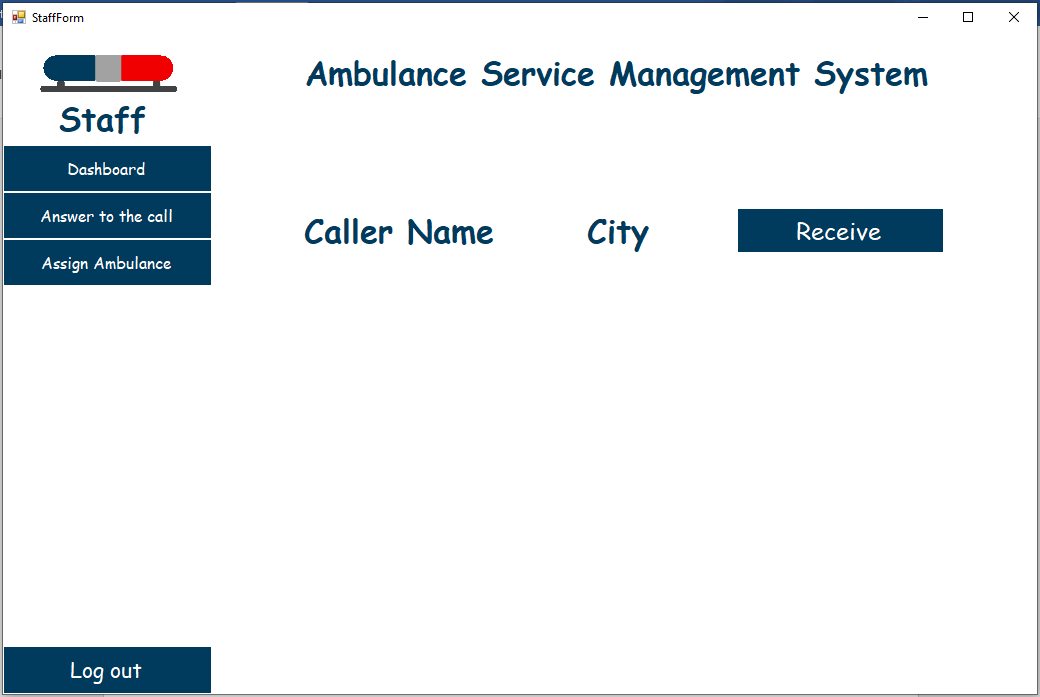


Figure 29: Answer to the Call (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| city\_label | Label | To show name of city and caller name. |
| caller\_name\_label |
| receive\_btn | Button | To receive emergency call from citizens. |

### Assign Ambulance Window (Call Receiver):

|  |  |
| --- | --- |
| Use case name | Assign Ambulance |
| Actor | Staff Member |
| Flow of events | Log in as a staff member 🡪 click on the Assign Ambulance button 🡪 click on the assign button |

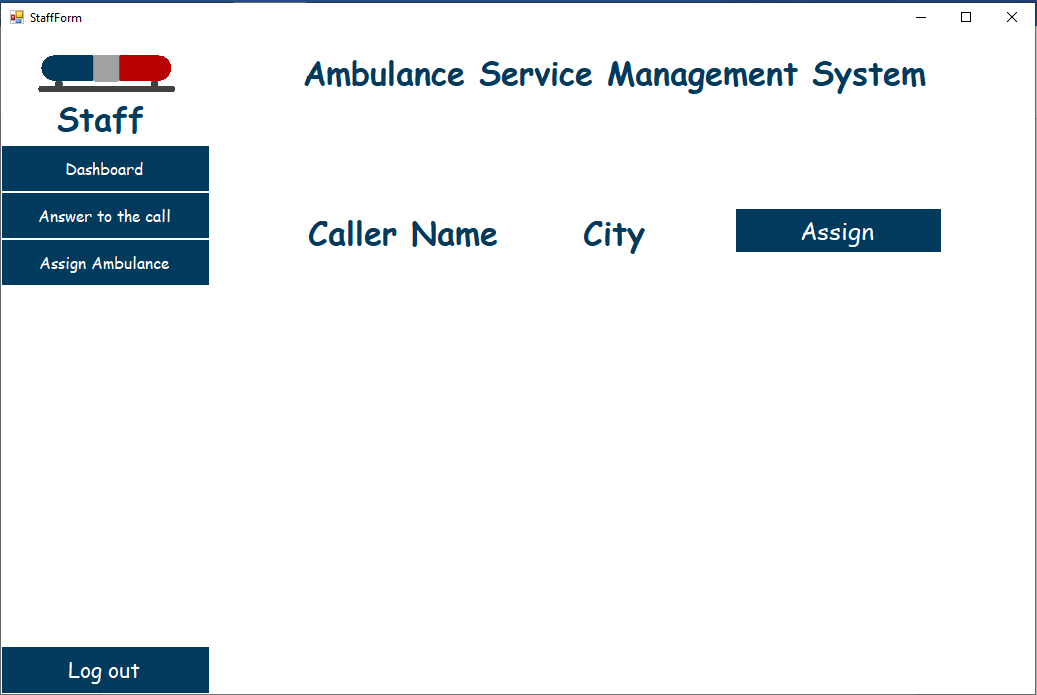


Figure 30: Assign Ambulance (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| city\_label | Label | To show text about name of city and caller name. |
| caller\_name\_label |
| assign\_btn | Button | To assign ambulance to the caller. |

### Call for Ambulance Window (Caller):

|  |  |
| --- | --- |
| Use case name | Make a Call |
| Actor | Caller |
| Flow of events | Click on the Call Ambulance button 🡪 click on the Provide info button 🡪 enter your name 🡪 select city 🡪 click on the Call button |

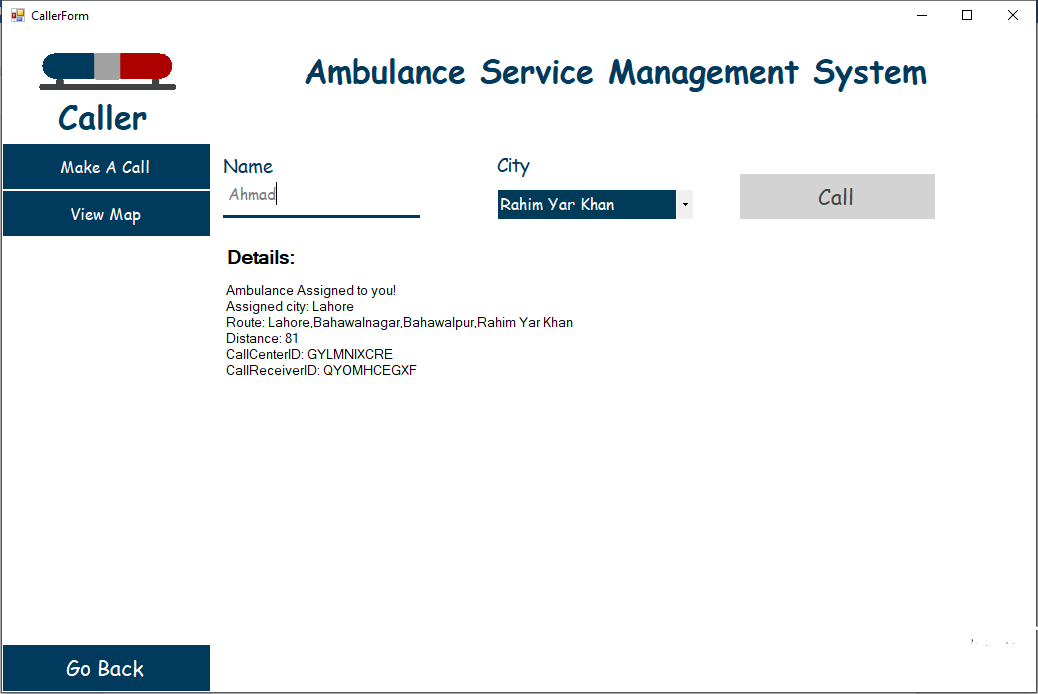


Figure 31: Call for Ambulance (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| city\_combo\_box | ComboBox | To select any city. |
| call\_btn | Button | To call for rescue. |
| name\_textbox | TextBox | To enter name. |
| diew\_details\_label | Label | To details about the assigned city, and the complete route from the caller city to the assigned city with the shortest path. |

### View Map Window (Admin):

|  |  |
| --- | --- |
| Use case name | View Map |
| Actor | Caller |
| Flow of events | Click on the Call Ambulance button 🡪 click on view map 🡪 enter patient name 🡪 enter CNIC number 🡪 enter your age 🡪 select your gender 🡪 click on submit button |

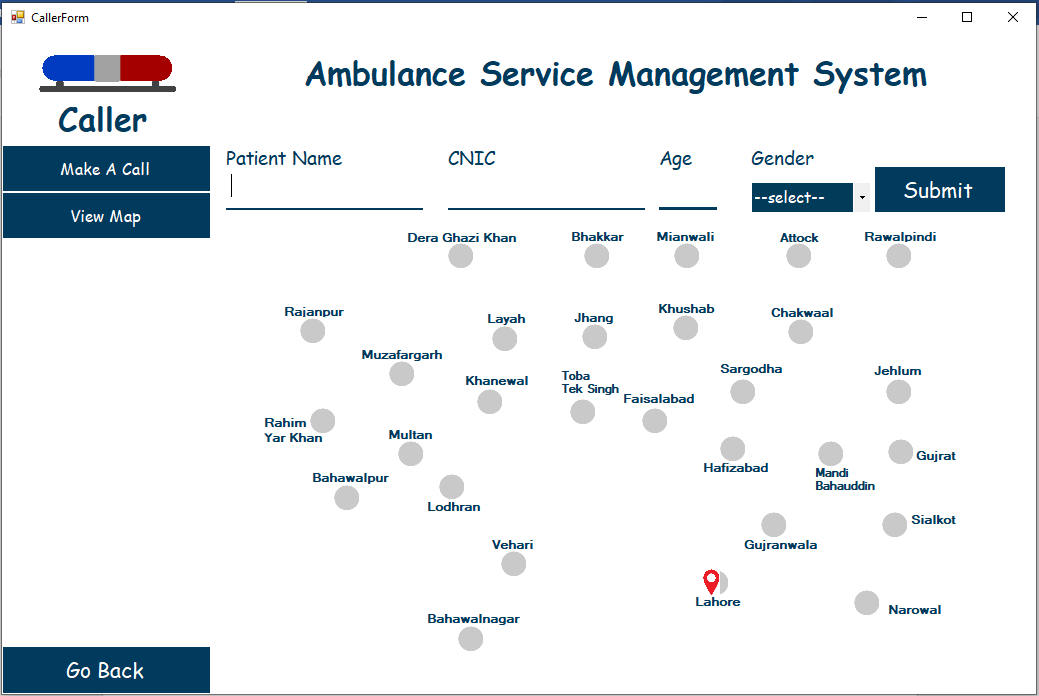


Figure 32: View Map (GUI)

|  |  |  |
| --- | --- | --- |
| UI Component Name | UI Component Type | Description |
| patient\_name\_textbox | TextBox | To enter patient name. |
| cnic\_textbox | To enter the CNIC of the patient. |
| age\_textbox | To enter the age of the patient. |
| submit\_btn | Button | To submit the details of the patient. |
| gender\_combo\_box | ComboBox | To select gender type. |
| view\_map\_panel | Panel | To show a map to visualize the status of the ambulance assigned to the caller |

# Complete Working:

Our App starts with a starting window, in which there is nothing special, then a login window is opened. Before the form opens, all of the cities, call centers, ambulances, call receivers, drivers, patients, and city edges data is loaded from the CSV files to their respective manager classes. As all of these manager classes are **singleton** then once their objects are created there will be no more objects created.

class SingletonClass:  
 static SingletonClass \_\_singletonClass 🡨 null  
 static SingletonClass getInstance():  
 if \_\_singletonClass == null:  
 \_\_ singletonClass 🡨 new SingletonClass()  
 return \_\_ singletonClass

In the **login window** user as an admin will enter its username and password. In the backend its username password if verified he/she will be redirected to the **admin window** where he/she will be able to view the dashboard and all of his/her other options.

In the **dashboard** panel, he/she will be able to see the number of call centers that are added and the total number of staff members and ambulances.

In the **Add Staff** panel, he/she will enter the name, phone, select gender, category, shift, city, enter the home address, and password. The ID of each staff will be randomly generated and he/she will not be allowed to change that ID. The cities in the select city combo box will be added on run time. Those cities will be added where call centers are provided. Then he/she will click on the add button and a new object of the staff member will be created. If the category of the staff member is call receiver, then a new object of the **CallReceiver** will be created and will be added to the list in the **CallReceiverManager** class. The same process is for the driver. After the data is added to the list then it will be written to the CSV file.

In the **View/UpdateStaff** panel, all of the staff members’ data will be shown to him/her. And also can be deleted and edited.

In the **AddHospital** panel, the name of the hospital and the number of beds are to be provided, and a city from city combo box is to be selected. As the add button is clicked, a new object of **CallCenter** is generated and a new id is created at the backend and is assigned to that call center. Then, that call center is added to the list in CallCenterManager class. Also, the id of that call center is assigned to an object of the **City** where the call center is to be added.

In the **View/UpdateHospital** and the **ViewPatients** panel, the data of the hospitals and patients will be shown and can be deleted and updated.

In the **AddAmbulance** panel, engine number, hospital and the purchase date of the ambulance s provided. The Id of the ambulance will be randomly generated and will be assigned to that ambulance. In the **ViewAmbulance** records of ambulances will be shown.

When a **caller** will make a call for ambulance his/her name and city be saved and **Dijkstra’s** algorithm will run from the **Graph** class to check for the nearest call center is available. Then the name and city of the caller is sent to that nearest call center’s call receiver and he/she will assign an ambulance to that caller.

In the **ViewMap** panel caller will be able to visualize the map.

public void Dijkstra(string start)

        {

            initializeVertices(start);

            List<CityVertex> queue = new List<CityVertex>();

            set = new List<CityVertex>();

            foreach (CityVertex cityVertex in vertices.Values)

            {

                queue.Add(cityVertex);

            }

            while(queue.Count > 0)

            {

                CityVertex cityVertex = queue[0];

                foreach(CityVertex c in queue)

                {

                    if (c.Distance < cityVertex.Distance)

                    {

                        cityVertex = c;

                    }

                }

                queue.Remove(cityVertex);

                set.Add(cityVertex);

                foreach (CityEdge cityEdge in adjLists[cityVertex])

                {

                    relax(cityVertex, cityEdge.Dest, cityEdge.Weight);

                }

            }

        }

        private void relax(CityVertex u, CityVertex v, int w)

        {

            if (v.Distance > u.Distance + w)

            {

                v.Distance = u.Distance + w;

                v.Parent = u;

            }

        }

        private void initializeVertices(string start)

        {

            foreach (CityVertex cityVertex in vertices.Values)

            {

                if(cityVertex.Name != start)

                {

                    cityVertex.Distance = Int32.MaxValue;

                    cityVertex.Parent = null;

                }

                else

                {

                    cityVertex.Distance = 0;

                }

            }

        }

# Flow Diagram:

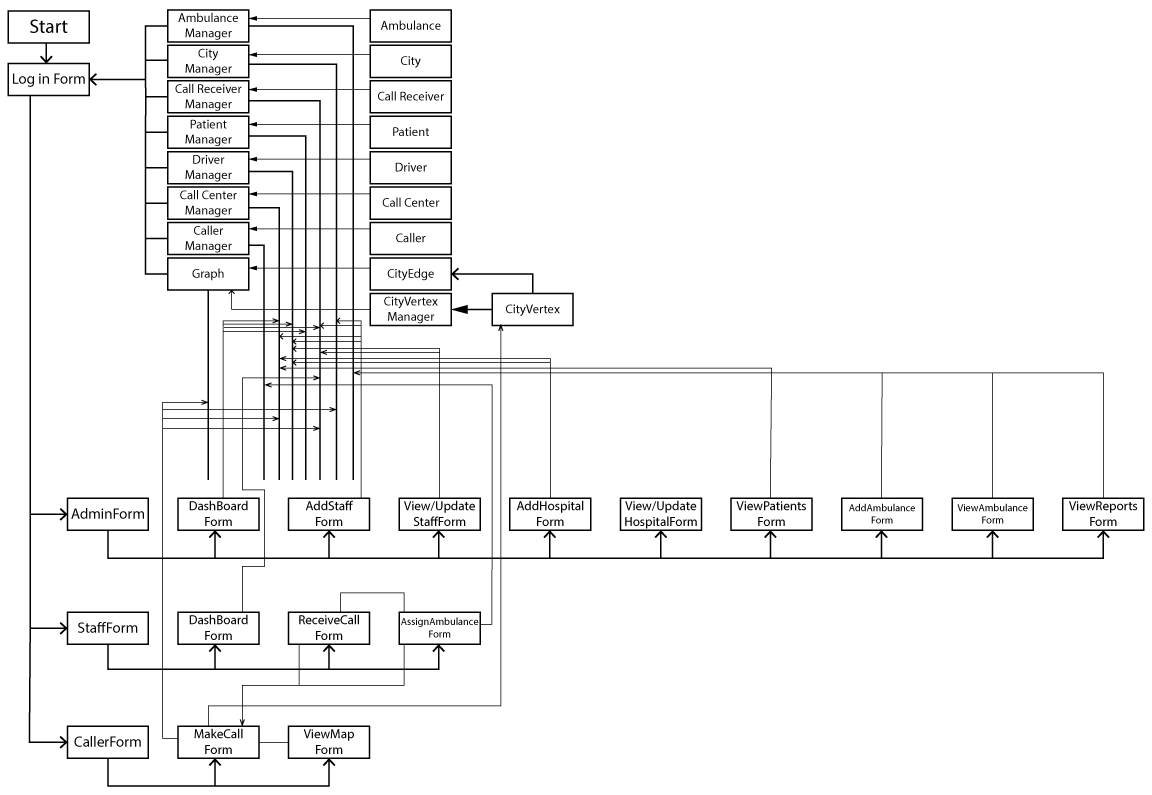


Figure 33: Flow diagram of complete project

# Class Diagrams:















