Aptech™ COMPUTER EDUCATION *Unleash your potential*

TECHWIZ EMPOWERING I.T.

Smart Traffic Management System

Shehryar Ahmed Subhani
Asim Fiaz
Nabeel Alam
Abdullah Junaid

Contents

Problem Definition

Urban cities across the world are frequently facing traffic issues. Main junctions inside cities encounter massive jams or bottlenecks in traffic, which result in frustration for vehicle owners, pedestrians, and city traffic department including traffic police and traffic controllers. This impacts day-to-day lives of all these individuals as well as emergency facilities. For instance, ambulances and fire brigades cannot afford to be stuck in such jams as every minute is precious to them to save lives.

Requirement Analysis

Users

Registration: The registration interface takes the following inputs from the user: User type, First Name, Last Name, Contact, E-Mail, Password, Address, and an image as proof. The Proof is an image that proves their identity of whatever user type they claim they are. As a pedestrian, they need only to provide an image of their CNIC, but as a police officer, they must provide an image of some official document that proves their profession. In order to avoid programmed bots spamming false registrations, a reCAPTCHA widget has also been added as part of the registration form. On proper completion of the registration form, a request will be submitted to the Admin and a notification will also be sent to them. The Admin can then view the registration request from the Admin Panel and accept or reject the request after seeing all details and the provided proof. In either case, an e-mail will be sent to the applying user telling them whether their application has been accepted or rejected.

Login: After a successful registration, a user can then use their e-mail and password to log in to the smart traffic management system. Once logged in, they can submit feedback, complaints, check complaint status, report emergencies through the different pages provided, and log out.

Notifications: A user will also receive notifications that have been issued by the police department about recent emergencies. A notification will have date, time, and a short description of the type of emergency. IMG INSERT SAMPLE NOTIFICATION IMAGE HERE

Submit Complaints: Users can easily submit any complaints that they wish for the traffic police to take a look at. In order to submit a complaint, a user must be registered and logged in, provide a title, type, and a description of their complaint. They must complete a reCAPTCHA in order for successful submission of their complaint.

Check Complaint Status: Once a user has submitted their complaint, they will be given a complaint ID which they can easily input into the complaint tracker and check the status of their complaint.

Emergency Help: In case of an emergency, a logged in user can quickly submit a form that only requests a title of their emergency, and completion of a reCAPTCHA widget to prove that they are human. Upon submission, the location of the user, along with their other details such as address and contact number are stored in the database, and notifications are dispatched to the officials so they can take appropriate actions.

Feedback: A registered user can submit their feedback regarding the state of the smart traffic management system via the feedback menu interface. They will need to be logged in, provide their feedback description, and prove their authenticity as a human user via the reCAPTCHA widget. Upon successful submission, the feedback will be sent to the application officials along with notification for said feedback.

Logout: A logged in user can easily press the logout button present in the top of every page in order to safely log out their account before they leave their device.

Traffic Control Department

Validate users: The department officials can view a user’s registration request, including full details and proof. Based on the provided information, they can then accept or reject the application. Registered users will have a unique 16-digit ID attached to them.

Create and send traffic notifications: The officials can create and send traffic notifications about heavy traffic, accidents, etc. to all registered users.

View and resolve complaints: The officials can view all complaints that have been sent in by the users. They can view the ID, title, description, and category of the complaint and mark them as solved.

View and send emergency help: Here, the officials will be able to see all details of any emergency that has been submitted by a user. They can view all details of the person that has submitted the emergency.

Check feedback: The officials can view any feedback sent by users in order to make improvements in the system.

Generate Reports: Here, the officials can easily generate daily reports such as a summary on complaints, feedback, etc.

1. Design Phase

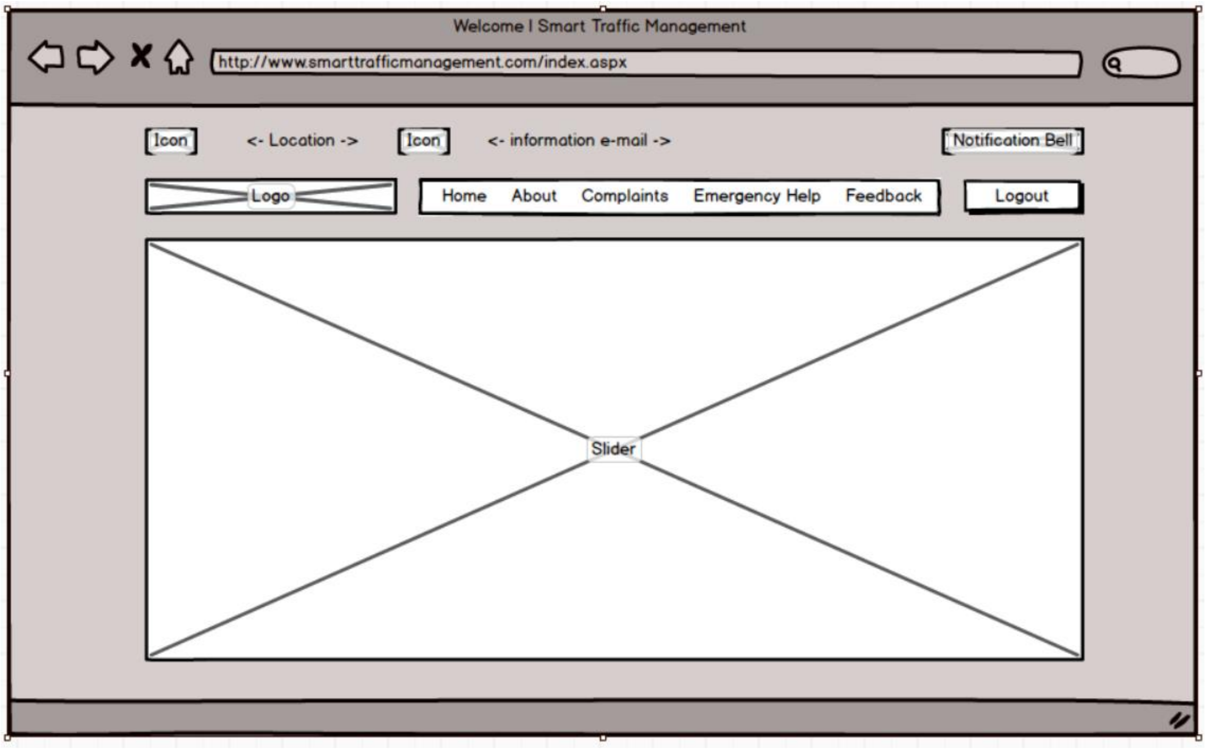


FIGURE 1: PUBLIC HOME / INDEX PAGE

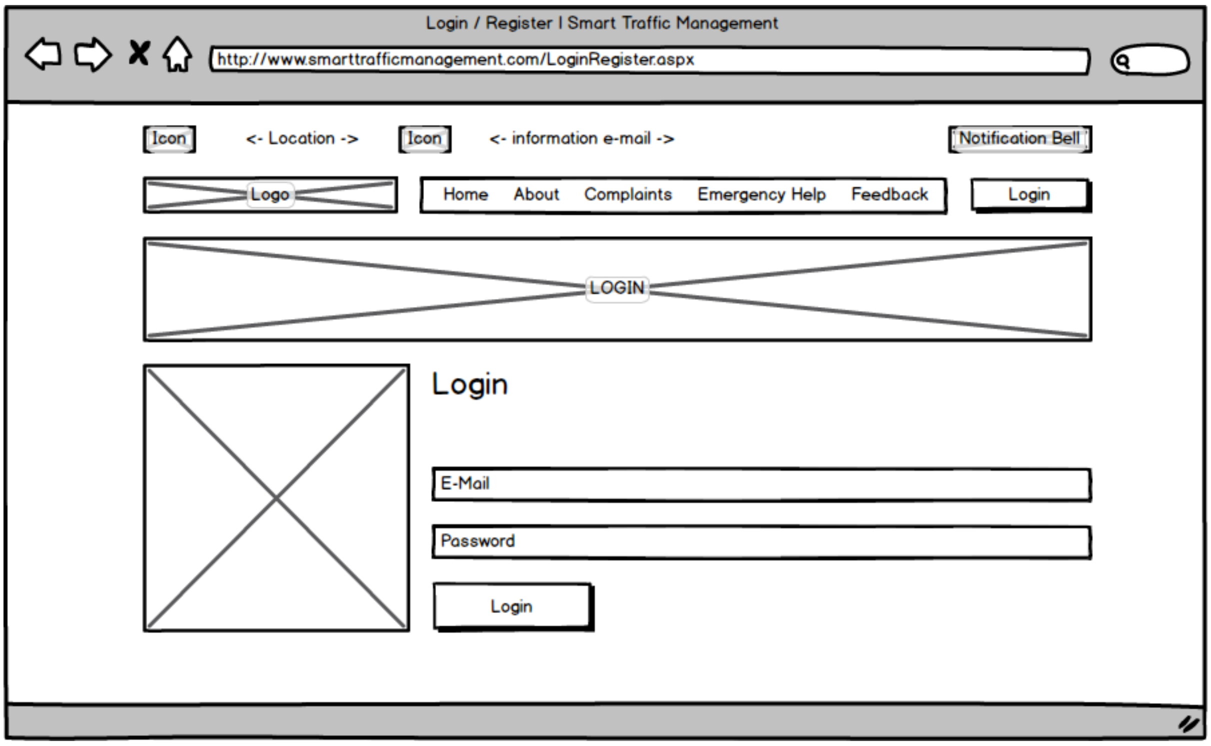


FIGURE 2: LOGIN PAGE

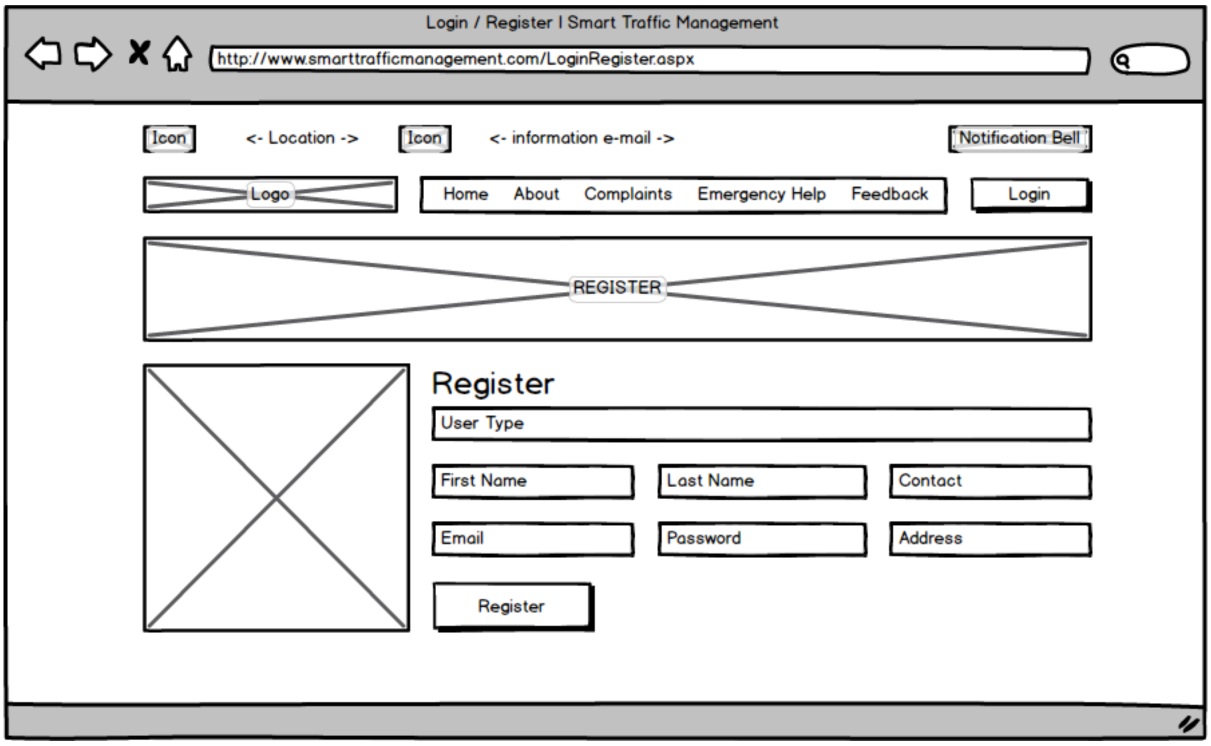


FIGURE 3: REGISTER PAGE

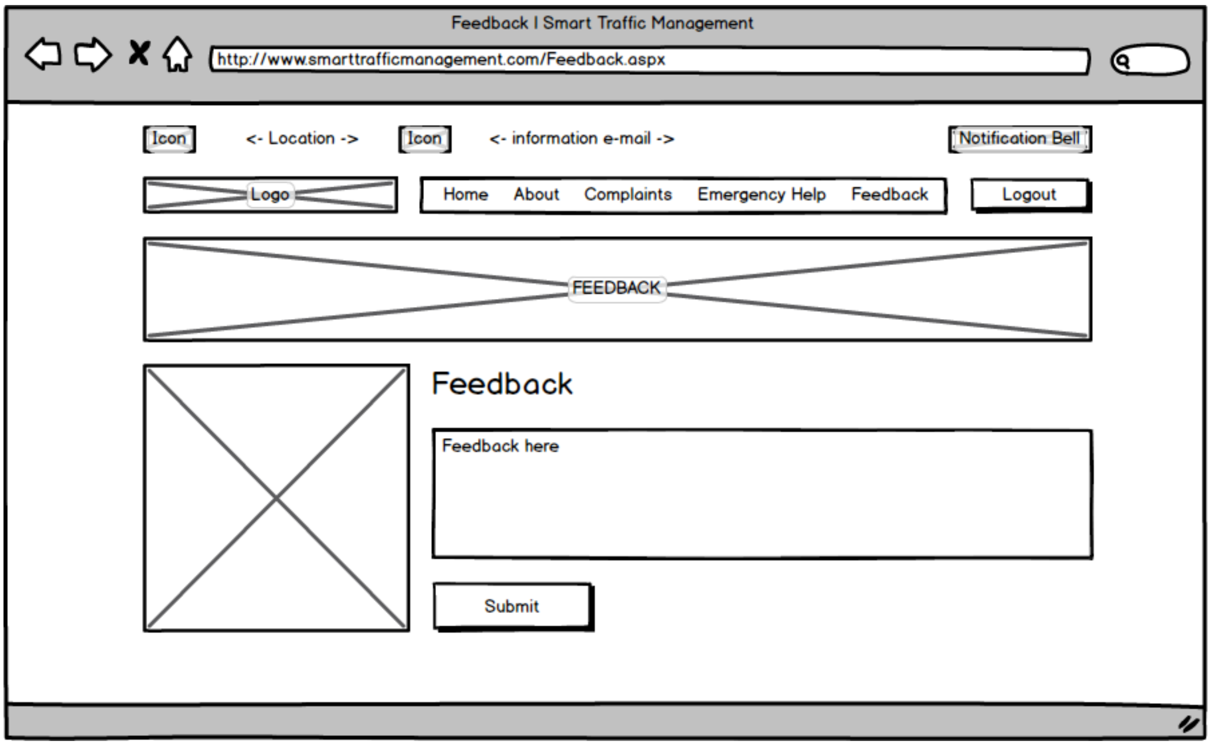


FIGURE 4: FEEDBACK PAGE

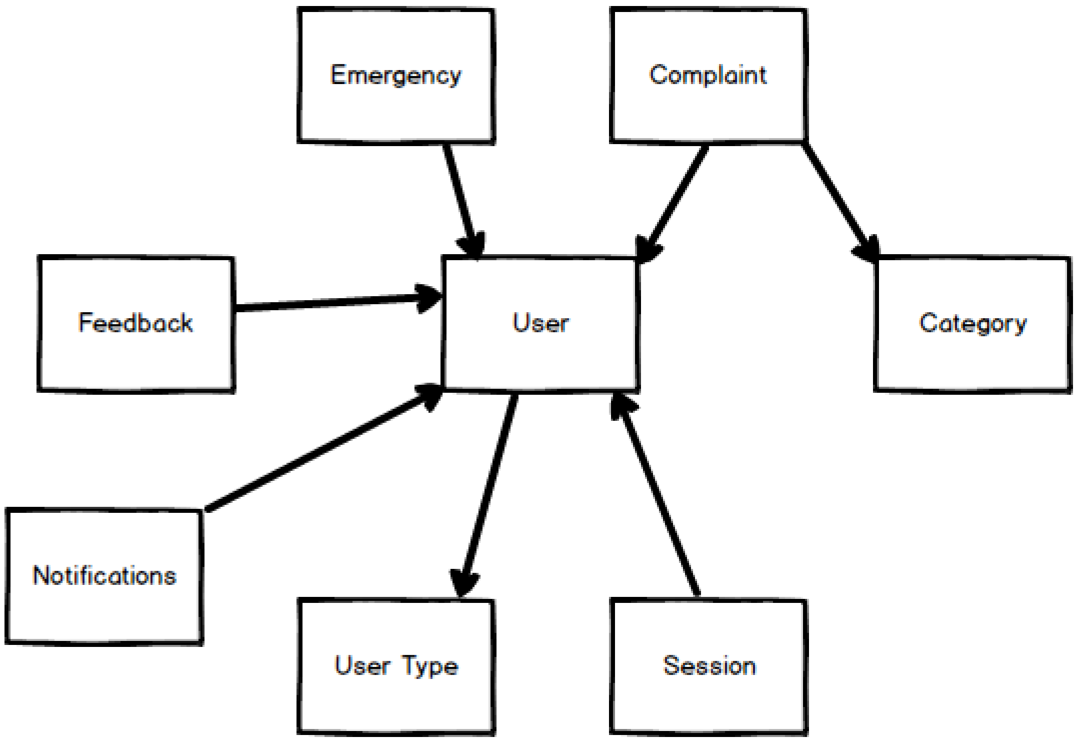
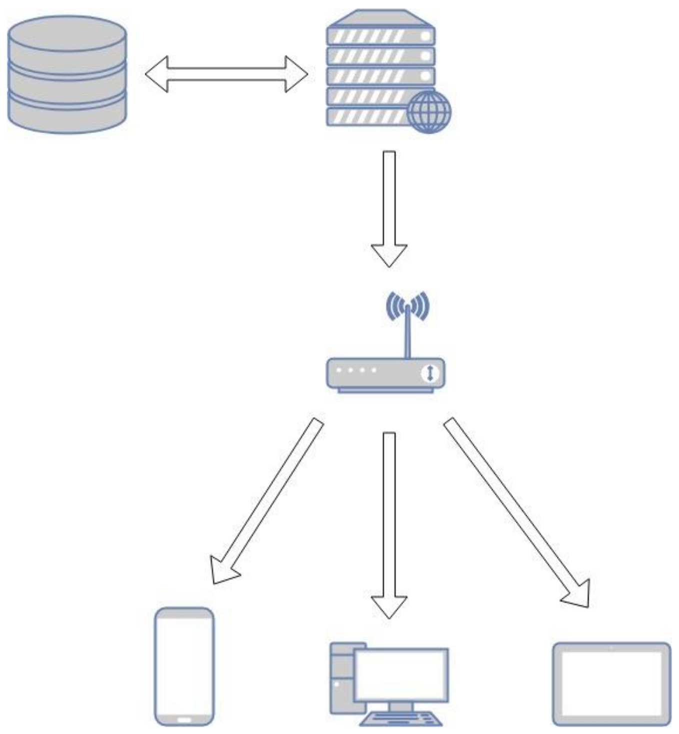
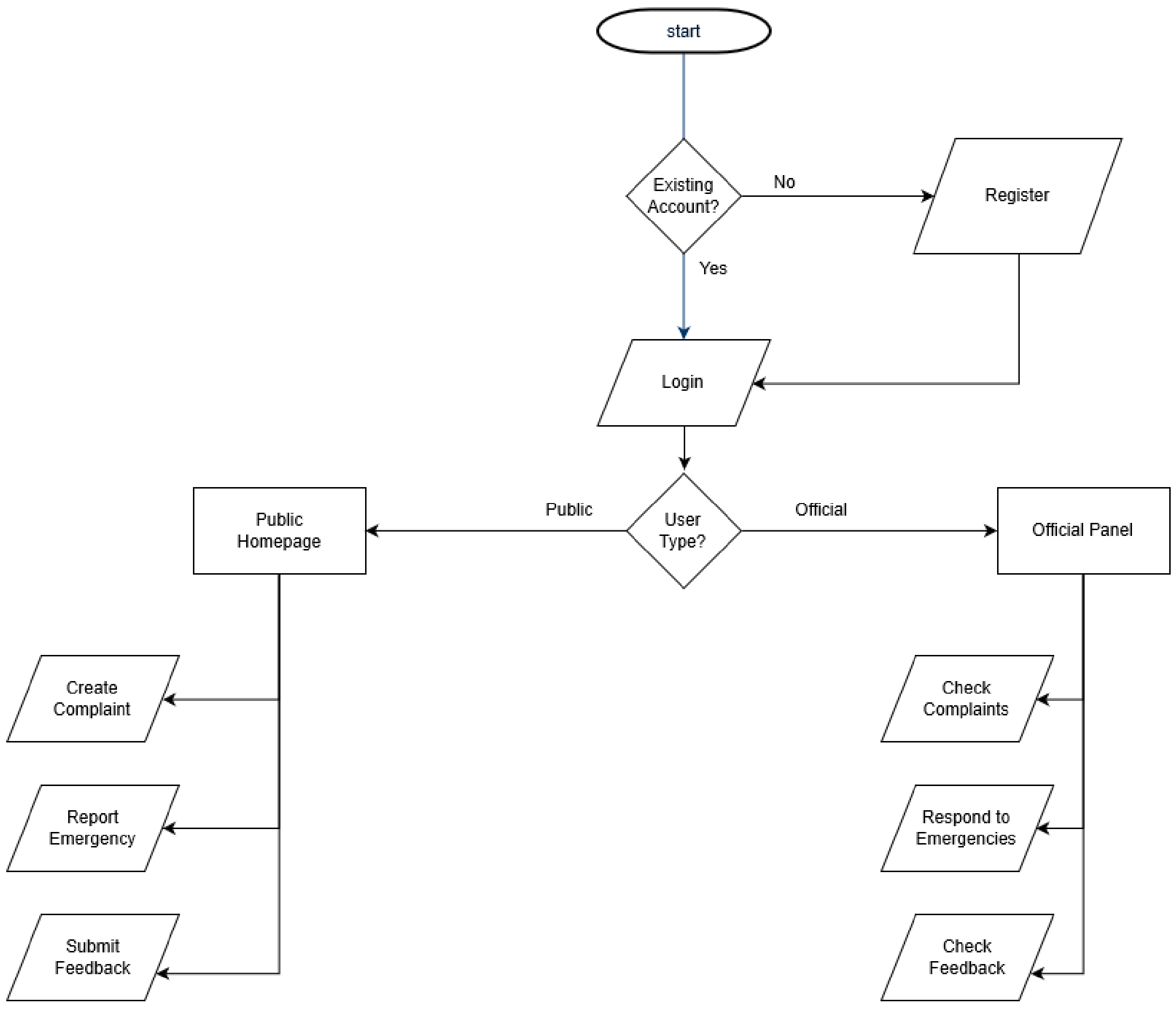


FIGURE 5: DATABASE

Application Architecture





Evaluation / Testing

The application was tested using false and improper input purposefully in order to break the code, creating exceptions. The generated exceptions were then taken care of, in order to leave behind an elegant application that does not break under live conditions.

Each member of the team was instructed by the project leader to test their code extensively to leave minimal margin for errors, bugs, and glitches as a form of black-box testing

Following that, normal users were brought in, who had no knowledge of the code and were made to surf the website to point out errors that developers did not consider.

As for evaluation, the code was lined up with project specifications to see if there was any requirement that was left out.

Project Tracking and Monitoring Activities

Design Plan: Asim Fiaz	Document Name: Project review	SWD/Form No. 11
Effective Date: 28/03/2019	Version: 1	Page No.

Date	Plan / Milestone	Work Specification	Status	Remarks	Responsibility
25/03/2019	SQL selection	Decided on which structured query language to use in the project	Completed	Microsoft SQL Server will be used	Shehryar Ahmed Subhani
25/03/2019	Software usage	Planning which software will be used by the team	Completed	Microsoft Visual Studio 2017 will be used	Nabeel Alam
25/03/2019	Methods and Variables naming convention decisions	All planning of which kind of naming conventions to be used throughout the project were finalized	Completed	Upper CamelCasing will be used for Methods, and lower camelCasing will be used for variables	Asim Fiaz
25/03/2019	First Mockups	Showcasing of first Mockups for the web application	Completed	None	Asim Fiaz
25/03/2019	Planning of front end and back end interface	Decided which template to use for the public front end and the Official back end of the application	Completed	None	Abdullah Junaid
26/03/2019	Finalization of the User Interface	Created a final look of the user interface which would be religiously followed by the team	Completed	None	Shehryar Ahmed Subhani
					Date: 28/03/2019

Documentation Section

Source Code

Website Master Page

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Forms/Public/public.Master"
AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="SmartTraffic.Index" %>

<asp:Content ID="Content2" ContentPlaceHolderID="cpTitle" runat="server">
    Welcome | Smart Traffic Management
</asp:Content>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

    <!-- Slider Start -->
    <div style="width: 100%; height: auto; margin: auto;">
        <section class="welcome-area">
            <div class="welcome-slides owl-carousel owl-loaded owl-drag">
                <div class="owl-stage-outer">
                    <div class="owl-stage" style="transform: translate3d(-3162px, 0px,
0px); transition: all 0s ease 0s; width: 6324px;"></div>
                    <div class="owl-item active" style="width: 100%;">
                        <div class="welcome-welcome-slide bg-img bg-gradient-overlay"
style="background-image: url(img/core-img/2.jpg);" data-jarallax-original-
styles="background-image: url(img/bg-img/2.jpg);">
                            <div class="contai4ner h-100">
                                <div class="row h-100 align-items-center">
                                    <div class="col-12">
                                        <!-- Welcome Text -->
                                        <div class="welcome-text text-center">
                                            <h2 data-animation="fadeInDown" data-
delay="100ms" style="animation-delay: 100ms; opacity: 1;" class="animated fadeInDown">
                                                We Believe Everyone Should Have<br
/>Easy Access To Smart Traffic
                                            </h2>
                                            <p data-animation="fadeInDown" data-
delay="300ms" style="animation-delay: 300ms; opacity: 1;" class="animated fadeInDown">
                                                As a leading industry innovator,
SmartTarffic is opening up
                                                exciting new opportunities for Rescue
professionals, &amp;
                                            </p>
                                        </div>
                                    </div>
                                </div>
                            </div>
                            <div id="jarallax-container-3" style="position: absolute;
top: 0px; left: 0px; width: 100%; height: 100%; overflow: hidden; pointer-events:
none; z-index: -100;">
                                <div style="background-position: 50% 50%; background-
size: cover; background-repeat: no-repeat; background-image:
url(&quot;file:///E:/dento/img/bg-img/2.jpg&quot;); position: absolute; top: 0px;
left: 0px; width: 1054px; height: 645.6px; overflow: hidden; pointer-events: none;
margin-top: 5.7px; transform: translate3d(0px, 1.5px, 0px);"></div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </section>
    </div>
    <!-- Slider End -->

    <div class="container">
        <div class="dento-border clearfix"></div>
    </div>

    <!-- Our Rescue Persons  -->
    <div class="container" style="padding-top: 20px;">
        <div class="row">
            <!-- Section Heading -->
            <div class="col-12">
                <div class="section-heading text-center">
                    <h2>Affiliated Officers</h2>
                    <div class="line"></div>
                </div>
            </div>
        </div>

        <div class="row">
            <!-- Single Officer Area -->
            <div class="col-12 col-sm-6 col-md-4">
                <div class="single-dentist-area mb-100">
                    <img src="img/core-img/police-chief.jpg" alt="">
                    <!-- Officer Content -->
                    <div class="dentist-content">
                        <!-- Social Info -->
                        <div class="dentist-social-info">
                            <a href="www.facebook.com"><i class="fa fa-
facebook"></i></a>
                            <a href="www.twitter.com"><i class="fa fa-
twitter"></i></a>
                            <a href="#"><i class="fa fa-google-plus"></i></a>
                        </div>
                        <!-- Officer Info -->
                        <div class="dentist-info bg-gradient-overlay">
                            <h5>Ulysha Renee</h5>
                            <p>Police Chief</p>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Single Officer Area -->
            <div class="col-12 col-sm-6 col-md-4">
                <div class="single-dentist-area mb-100">
                    <img src="img/core-img/doctor.jpg" alt="">
                    <!-- Officer Content -->
                    <div class="dentist-content">
                        <!-- Social Info -->
                        <div class="dentist-social-info">
                            <a href="www.facebook.com"><i class="fa fa-
facebook"></i></a>
                            <a href="www.twitter.com"><i class="fa fa-
twitter"></i></a>
                            <a href="#"><i class="fa fa-google-plus"></i></a>
                        </div>
                        <!-- Officer Info -->
                        <div class="dentist-info bg-gradient-overlay">
                            <h5>Dr. Siddhart</h5>
                            <p>Medical Expert</p>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Single Officer Area -->
            <div class="col-12 col-sm-6 col-md-4">
                <div class="single-dentist-area mb-100">
                    <img src="img/core-img/11.jpg" alt="">
                    <!-- Officer Content -->
                    <div class="dentist-content">
                        <!-- Social Info -->
                        <div class="dentist-social-info">
                            <a href="www.facebook.com"><i class="fa fa-
facebook"></i></a>
                            <a href="www.twitter.com"><i class="fa fa-
twitter"></i></a>
                            <a href="#"><i class="fa fa-google-plus"></i></a>
                        </div>
                        <!-- Officer Info -->
                        <div class="dentist-info bg-gradient-overlay">
                            <h5>Albert Hulk</h5>
                            <p>Rescue Commander</p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <!-- Our Rescue Persons End -->

    <!-- Testimonals Start -->
    <section class="dento-cta-area">
        <div class="container">
            <div class="row">
                <!-- Cool Facts Area -->
                <div class="col-12 col-sm-6 col-lg-3">
                    <div class="single-cta text-center mt-50 mb-100">
                        <i class="icon_genius"></i>
                        <h2><span class="counter">20</span></h2>
                        <h5>Countries covered</h5>
                    </div>
                </div>

                <!-- Cool Facts Area -->
                <div class="col-12 col-sm-6 col-lg-3">
                    <div class="single-cta text-center mt-50 mb-100">
                        <i class="icon_heart_alt"></i>
                        <h2><span class="counter">7000</span>+</h2>
                        <h5>Happy Customers</h5>
                    </div>
                </div>

                <!-- Cool Facts Area -->
                <div class="col-12 col-sm-6 col-lg-3">
                    <div class="single-cta text-center mt-50 mb-100">
                        <i class="icon_book_alt"></i>
                        <h2><span class="counter">1200</span>+</h2>
                        <h5>licences Issued</h5>
                    </div>
                </div>
                <!-- Cool Facts Area -->
                <div class="col-12 col-sm-6 col-lg-3">
                    <div class="single-cta text-center mt-50 mb-100">
                        <i class="icon_id"></i>
                        <h2><span class="counter">40</span>+</h2>
                        <h5>Experts</h5>
                    </div>
                </div>
            </div>
        </div>
    </section>
    <!-- Testimonals End -->

</asp:Content>
```

Portal Source Code

```
<%@ Page Title="" Language="C#" MasterPageFile="admin.Master" AutoEventWireup="true"
CodeBehind="Index.aspx.cs" Inherits="SmartTraffic.Forms.Admin.Index" %>

<asp:Content ID="Content3" ContentPlaceHolderID="cpTitle" runat="server">
    Dashboard | Smart Traffic Management
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceholder" runat="server">
    <div class="col-xs-12">
        <div class="page-title-box">
            <h4 class="page-title">Emergencies</h4>
            <ol class="breadcrumb p-0 m-0">
                <li>SmartTraffic
                </li>
                <li class="active">Dashboard
                </li>
            </ol>
            <div class="clearfix"></div>
        </div>
    </div>
    <div class="row">
        <!-- Visitors today -->
        <div class="col-lg-3">
            <div class="card-box widget-box-one" style="background-color: blue; color:
white">
                <i class="mdi mdi-chart-areaspline widget-one-icon"></i>
                <div class="wigdet-one-content">
                    <p title="Website Hits Today" class="m-0 text-uppercase font-600
font-secondary text-overflow">
                        Website Hits Today
                    </p>
                    <h2>
                        <asp:Label runat="server" ID="lblWebsiteHitsToday"
ForeColor="White"></asp:Label>
                    </h2>
                    <p class="text-muted m-0">
                        <b style="color: white">Last:</b>
                        <asp:Label runat="server" ID="lblWebsiteHitsLast"
ForeColor="White"></asp:Label><span style="color: white"> hits</span>
                    </p>
                </div>
            </div>
        </div>

        <!-- Feedbacks Today -->
        <div class="col-lg-3">
            <div class="card-box widget-box-one" style="background-color: orangered;
color: white">
                <i class="mdi mdi-thumb-up-outline widget-one-icon"></i>
                <div class="wigdet-one-content">
                    <p title="Feedbacks Today" class="m-0 text-uppercase font-600
font-secondary text-overflow">
                        Feedbacks Today
                    </p>
                    <h2>
                        <asp:Label runat="server" ID="lblFeedbacksToday" style="color:
white"></asp:Label>
                    </h2>
                    <p class="text-muted m-0" style="color: white">
                        <b style="color: white">Last:</b>
                        <asp:Label runat="server" ID="lblFeedbacksLast" style="color:
white"></asp:Label>
                        <span style="color: white"> feedbacks</span>
                    </p>
                </div>
            </div>
        </div>

        <!-- Complaints Today -->
        <div class="col-lg-3">
            <div class="card-box widget-box-one" style="background-color: green;
color: white">
                <i class="mdi mdi-clipboard-outline widget-one-icon"></i>
                <div class="wigdet-one-content">
                    <p title="Complaints Today" class="m-0 text-uppercase font-600
font-secondary text-overflow">
                        Complaints Today
                    </p>
                    <h2>
                        <asp:Label runat="server" ID="lblComplaintsToday"
style="color: white"></asp:Label>
                    </h2>
                    <p class="text-muted m-0">
                        <b style="color: white">Last:</b>
                        <asp:Label runat="server" ID="lblComplaintsLast" style="color:
white"></asp:Label>
                        <span style="color: white">complaints</span>
                    </p>
                </div>
            </div>
        </div>

        <!-- Emergencies Today -->
        <div class="col-lg-3">
            <div class="card-box widget-box-one" style="background-color: darkred;
color: white">
                <i class="mdi mdi-chart-areaspline widget-one-icon"></i>
                <div class="wigdet-one-content">
                    <p title="Emergencies Today" class="m-0 text-uppercase font-600
font-secondary text-overflow">
                        Emergencies Today
                    </p>
                    <h2>
                        <asp:Label runat="server" ID="lblEmergenciesToday"
style="color: white"></asp:Label>
                    </h2>
                    <p class="text-muted m-0">
```

```
                        <b style="color: white">Last:</b>
                        <asp:Label runat="server" ID="lblEmergenciesLast"
style="color: white"></asp:Label>
                        <span style="color: white">emergencies</span>
                    </p>
                </div>
            </div>
        </div>
    </div>
    <div class="row">
        <!-- Recent Feedback -->
        <div class="col-lg-6">
            <div class="card-box">
                <h4 class="header-title m-t-0 m-b-30">Recent Users</h4>
                <div class="table-responsive">
                    <table class="table table table-hover m-0">
                        <thead>
                            <tr>
                                <th></th>
                                <th>Name</th>
                                <th>Contact</th>
                                <th>Date</th>
                            </tr>
                        </thead>
                        <tbody class=".table-striped">
                            <asp:Repeater runat="server" ID="rUsers"
EnableViewState="true">
                                <ItemTemplate>
                                    <tr style="height: 2em; line-height: 2em;">
                                        <asp:TableCell runat="server"
ID="tcIcon"><span class="avatar-sm-box bg-<%= SmartTraffic.Utilities.getRandomClass()
%>"><%# Eval("UserFName").ToString().First() %></span></asp:TableCell>
                                        <asp:TableCell runat="server" ID="tcUserName"
Text='<%# Eval("UserFName") + " " + Eval("UserLName") %>' Width="30%" Style="vertical-
align: middle;"></asp:TableCell>
                                        <asp:TableCell runat="server" ID="tcContact"
Text='<%# Eval("UserContact") %>' Width="30%" Style="vertical-align:
middle;"></asp:TableCell>
                                        <asp:TableCell runat="server" ID="tcDate"
Text='<%# Eval("UserRegDateTime") %>' Width="35%" Style="vertical-align:
middle;"></asp:TableCell>

                                    </tr>
                                </ItemTemplate>
                            </asp:Repeater>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>

        <!-- Complaints -->
        <div class="col-lg-6">
            <div class="card-box">
                <h4 class="header-title m-t-0 m-b-30">Recent Complaints</h4>
                <div class="table-responsive">
                    <table class="table table table-hover m-0">
                        <thead>
                            <tr>
                                <th>Title</th>
                                <th>Category</th>
                                <th>Status</th>
                            </tr>
                        </thead>
                        <tbody class=".table-striped">
                            <asp:Repeater runat="server" ID="rComplaints"
EnableViewState="true">
                                <ItemTemplate>
                                    <tr style="height: 3.5em; line-height: 3.5em;">
                                        <asp:TableCell runat="server" ID="tcTitle"
Text='<%# Eval("ComplaintTitle") %>' Width="40%" Style="vertical-align:
middle;"></asp:TableCell>
                                        <asp:TableCell runat="server" ID="tcCategory"
Text='<%# Eval("CategoryName") %>' Width="30%" Style="vertical-align:
middle;"></asp:TableCell>
                                        <asp:TableCell runat="server" ID="tcStatus"
Text='<%# Eval("ComplaintStatus") %>' Width="30%" Style="vertical-align:
middle;"></asp:TableCell>

                                    </tr>
                                </ItemTemplate>
                            </asp:Repeater>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
    <div class="row">
        <!-- Emergencies -->
        <!-- ? -->
    </div>
</asp:Content>
```

ControlManager.cs

```
public static class ControlManager
    {

        /*
         * Inputs the DatabaseContext class required, the name of the method in that
class and the name of the Repeater that needs to be populated
         * Populates the Repeater
         * No output
         */

        public static void populateControl(object db, string method, Repeater
repeater)
        {

            Type type = db.GetType();                                   //Gets the
type of the specified object
            PropertyInfo propertyInfo = type.GetProperty(method);       //Gets
specified property from the object
            object values = propertyInfo.GetValue(db, null);            //Gets value
at the specified property

            repeater.DataSource = values;                               //Populates
Repeater
            repeater.DataBind();

        }

        /*
```

```
         * Inputs the list which is populated with the data to populate the Repeater
and the name of the Repeater that needs to be populated
         * Populates the GridView
         * No output
         */

        // Use if the entire table is not required to be fetched
        public static void populateControl(object list, Repeater repeater)
        {
            repeater.DataSource = list;                              //Populates
Repeater
            repeater.DataBind();
        }

        /*
         * Inputs the list which is populated with the data to populate the
DropDownList and the name of the Repeater that needs to be populated
         * Populates the DropDownList
         * No output
         */

        public static void populateControl(object list, DropDownList dropDownList,
string dataTextField, string dataValueField)
        {

            dropDownList.DataTextField = dataTextField;
            dropDownList.DataValueField = dataValueField;

            dropDownList.DataSource = list;                          //Populates
DropDownList
            dropDownList.DataBind();

        }

        /*
         * Inputs a View
         * Finds all controls of type TextBox in that view and clears their text
         * No output
         */

        public static void clearTextBoxes(View view)
        {
            foreach (TextBox textBox in view.Controls.OfType<TextBox>())
            {
                textBox.Text = null;
            }
        }

        /*
         * Inputs a MultiView
         * Finds all controls of type TextBox in each of its Views and hides all the
panels
         * No output
         */

        public static void hidePanels(MultiView multiView)
        {
            foreach (View view in multiView.Controls.OfType<View>())
            {
                foreach (Panel panel in view.Controls.OfType<Panel>())
                {
                    panel.Visible = false;
                }
            }
        }
    }
```

EmailManager.cs

```
public static class EmailManager
    {
        public static void SendEmail(tblUser user)
        {
            MailMessage message = new MailMessage();
            SmtpClient client = new SmtpClient();
            client.Host = "smtp.gmail.com";
            client.Port = 587;

            message.From = new MailAddress("smarttraffic123@gmail.com");
            message.To.Add(user.UserEmail);
            message.Subject = "Account Verification Email";
            message.Body = "Dear " + user.UserFName + " " + user.UserLName +
",<br><br>Your account has successfully been verified!<br><br>Regards,<br>Smart
Traffic Management Team";
            message.IsBodyHtml = true;

            client.EnableSsl = true;
            client.UseDefaultCredentials = true;
            client.Credentials = new
System.Net.NetworkCredential("smarttraffic123@gmail.com", "@smart123");
            client.Send(message);
        }
    }

[System.Security.SecurityCritical]
    public static class LocationManager
    {
        public static void getLocation(int sessionID)
        {

            CLocation myLocation = new CLocation();
            myLocation.GetLocationEvent(sessionID);

        }

        public static double getLongitude(int EmergencyID)
        {
            using(DatabaseDataContext db = new DatabaseDataContext())
            {
                tblEmergency emergency = db.tblEmergencies.Where(x => x.EmergencyID ==
EmergencyID).FirstOrDefault();

                return emergency.EmergencyLongitude;
            }
        }

        public static double getLatitude(int EmergencyID)
        {
            using (DatabaseDataContext db = new DatabaseDataContext())
            {
                tblEmergency emergency = db.tblEmergencies.Where(x => x.EmergencyID ==
EmergencyID).FirstOrDefault();

                return emergency.EmergencyLatitude;
            }
        }

        private class CLocation
        {
            private GeoCoordinateWatcher watcher;
            private int SessionID;

            public void GetLocationEvent(int sessionID)
            {
                watcher = new GeoCoordinateWatcher();
                watcher.PositionChanged += new 
EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);

                SessionID = sessionID;

                watcher.TryStart(false, TimeSpan.FromMilliseconds(5000));
            }

            private void watcher_PositionChanged(object sender, 
GeoPositionChangedEventArgs<GeoCoordinate> e)
            {
                saveLocation(e.Position.Location.Latitude, 
e.Position.Location.Longitude);
            }

            private void saveLocation(double Latitude, double Longitude)
            {
                using(DatabaseDataContext db = new DatabaseDataContext())
                {
                    tblSession session = db.tblSessions.Where(x => x.SessionID == 
SessionID).FirstOrDefault();

                    if(session.SessionLongitude == 0)
                    {
                        session.SessionLongitude = Longitude;
                    }

                    if (session.SessionLatitude == 0)
                    {
                        session.SessionLatitude = Latitude;
                    }

                    db.SubmitChanges();
                }
            }
        }
    }

public class NotificationManager
    {
        public static void requestSent()
        {
            using (DatabaseDataContext db = new DatabaseDataContext())
            {

                tblNotification notification = new tblNotification
                {
                    NotificationMessage = "An account verification request has been 
received!",
                    NotificationLink = "Requests.aspx",
                    NotificationDateTime = DateTime.UtcNow,
                    NotificationType = "A",
                    UserID = null,
                    IsSeen = false
                };

                db.tblNotifications.InsertOnSubmit(notification);

                //Check
                db.SubmitChanges();
            }

        }

        public static void complaintSent()
        {
            using (DatabaseDataContext db = new DatabaseDataContext())
            {

                tblNotification notification = new tblNotification
                {
                    NotificationMessage = "A new complaint has been received!",
                    NotificationLink = "Complaints.aspx",
                    NotificationDateTime = DateTime.UtcNow,
                    NotificationType = "C",
                    UserID = null,
                    IsSeen = false
                };

                db.tblNotifications.InsertOnSubmit(notification);

                //Check
                db.SubmitChanges();
            }

        }

        public static void emergencySent()
        {
            using (DatabaseDataContext db = new DatabaseDataContext())
            {

                tblNotification notification = new tblNotification
                {
                    NotificationMessage = "An emergency has been received",
                    NotificationLink = "Emergencies.aspx",
                    NotificationDateTime = DateTime.UtcNow,
                    NotificationType = "E",
                    UserID = null,
                    IsSeen = false
                };

                db.tblNotifications.InsertOnSubmit(notification);

                //Check
                db.SubmitChanges();
            }

        }

        public static void feedbackSent()
        {
            using (DatabaseDataContext db = new DatabaseDataContext())
```

```
            {

                tblNotification notification = new tblNotification
                {
                    NotificationMessage = "New feedback has been received!",
                    NotificationLink = "Feedback.aspx",
                    NotificationDateTime = DateTime.UtcNow,
                    NotificationType = "F",
                    UserID = null,
                    IsSeen = false
                };

                db.tblNotifications.InsertOnSubmit(notification);

                //Check
                db.SubmitChanges();
            }
        }

        public static void alertSent(string description)
        {
            using (DatabaseDataContext db = new DatabaseDataContext())
            {
                List<tblUser> users = db.tblUsers.Where(x => x.UserTypeID != 1 &&
x.IsVerified == true && x.UserTypeID != 2).ToList();

                foreach (tblUser user in users)
                {
                    tblNotification notification = new tblNotification
                    {
                        NotificationDateTime = DateTime.UtcNow,
                        NotificationLink = "#",
                        NotificationMessage = description,
                        NotificationType = "B"
                    };

                    db.tblNotifications.InsertOnSubmit(notification);
                    db.SubmitChanges();
                }
            }
        }

        // Gets the number of unread notifications by a user
        public static string unseenNotificationCountAdmin()
        {
            using (DatabaseDataContext db = new DatabaseDataContext())
            {
                //int count = db.tblNotifications.Where(x => x.UserID ==
SessionManager.UserID && x.IsSeen == false).ToList().Count;

                int count = db.tblNotifications.Where(x => !x.UserID.HasValue &&
x.IsSeen == false).ToList().Count;

                return count.ToString();
            }

        }

        // Sets the icon displayed alongside the notification based on the type of
notification
        public static string notificationTimeElapsed(DateTime dateTime)
        {
            TimeSpan timeSpan = DateTime.UtcNow - dateTime;

            MidpointRounding mode = MidpointRounding.ToEven;

            return Math.Round(timeSpan.TotalSeconds, mode) < 60
                ? "A few seconds ago"
                : Math.Round(timeSpan.TotalMinutes, mode) >= 1 &&
Math.Round(timeSpan.TotalMinutes, mode) < 60
                    ? Math.Round(timeSpan.TotalMinutes, mode) == 1
                                    ? "1 minute ago"
                                    : Math.Round(timeSpan.TotalMinutes,
mode).ToString() + " minutes ago"
                    : Math.Round(timeSpan.TotalHours, mode) >= 1 &&
Math.Round(timeSpan.TotalHours, mode) < 24
                                    ? Math.Round(timeSpan.TotalHours, mode) == 1 ? "An
hour ago" : Math.Round(timeSpan.TotalHours, mode) + " hours ago"
                                    : Math.Round(timeSpan.TotalDays, mode) >= 1 &&
Math.Round(timeSpan.TotalDays, mode) <= 7
                                                    ? Math.Round(timeSpan.TotalDays,
mode) == 1
                                                                    ? "Yesterday"
                                                                    :
Math.Round(timeSpan.TotalDays, mode) == 7 ? "A week ago" :
dateTime.DayOfWeek.ToString()
                                                    :
dateTime.Date.ToLongDateString();
        }

        //Gets the difference from when the notification was created till the present
time
        public static string notificationType(string type)
        {
            type = type.Trim();

            switch (type)
            {
                case "A":
                    return "account-plus";
                case "B":
                    return "exclamation";
                case "C":
                    return "alert-circle";
                case "E":
                    return "ambulance";
                case "F":
                    return "thumbs-up-down";
                default:
                    return null;
            }
        }
    }
```

ReCaptha.cs

```
public class ReCaptcha {
        public static string Validate(string EncodedResponse) {
            var client = new System.Net.WebClient();

            string PrivateKey = "6Lf-WJoUAAAAAE_TYqgEWtYNvJhRa6nzmjrZbshP";

            var GoogleReply =
client.DownloadString(string.Format("https://www.google.com/recaptcha/api/siteverify?s
ecret={0}&response={1}", PrivateKey, EncodedResponse));
```

```
            var captchaResponse =
JsonConvert.DeserializeObject<ReCaptcha>(GoogleReply);

            return captchaResponse.Success.ToLower();
        }

        [JsonProperty("success")]
        public string Success {
            get { return m_Success; }
            set { m_Success = value; }
        }

        private string m_Success;
        [JsonProperty("error-codes")]
        public List<string> ErrorCodes {
            get { return m_ErrorCodes; }
            set { m_ErrorCodes = value; }
        }

        private List<string> m_ErrorCodes;
    }
```

SecurityManager.cs

```
public class SecurityManager
    {

        // Copied from https://stackoverflow.com/questions/10168240/encrypting-
decrypting-a-string-in-c-sharp as using AES encryption algorithms is best practice.

        private const int Keysize = 256;
        private const int iterations = 1000;

        public static string encrypt(string plainText, string passPhrase)
        {
            var saltString = generate256BitsOfRandomEntropy();
            var ivString = generate256BitsOfRandomEntropy();
            var plainTextBytes = Encoding.UTF8.GetBytes(plainText);

            using (var password = new Rfc2898DeriveBytes(passPhrase, saltString,
iterations))
            {
                var keyBytes = password.GetBytes(Keysize / 8);
                using (var symmetricKey = new RijndaelManaged())
                {
                    symmetricKey.BlockSize = 256;
                    symmetricKey.Mode = CipherMode.CBC;
                    symmetricKey.Padding = PaddingMode.PKCS7;
                    using (var encryptor = symmetricKey.CreateEncryptor(keyBytes,
ivString))
                    {
                        using (var memoryStream = new MemoryStream())
                        {
                            using (var cryptoStream = new CryptoStream(memoryStream,
encryptor, CryptoStreamMode.Write))
                            {
                                cryptoStream.Write(plainTextBytes, 0,
plainTextBytes.Length);
                                cryptoStream.FlushFinalBlock();


                                var cipherTextBytes = saltString;
                                cipherTextBytes =
cipherTextBytes.Concat(ivString).ToArray();
                                cipherTextBytes =
cipherTextBytes.Concat(memoryStream.ToArray()).ToArray();
                                memoryStream.Close();
                                cryptoStream.Close();
                                return Convert.ToBase64String(cipherTextBytes);
                            }
                        }
                    }
                }
            }
        }

        public static string decrypt(string cipherText, string passPhrase)
        {
            var cipherTextBytesWithSaltAndIv = Convert.FromBase64String(cipherText);

            var saltStringBytes = cipherTextBytesWithSaltAndIv.Take(Keysize /
8).ToArray();

            var ivStringBytes = cipherTextBytesWithSaltAndIv.Skip(Keysize /
8).Take(Keysize / 8).ToArray();

            var cipherTextBytes = cipherTextBytesWithSaltAndIv.Skip((Keysize / 8) *
2).Take(cipherTextBytesWithSaltAndIv.Length - ((Keysize / 8) * 2)).ToArray();

            using (var password = new Rfc2898DeriveBytes(passPhrase, saltStringBytes,
iterations))
            {
                var keyBytes = password.GetBytes(Keysize / 8);
                using (var symmetricKey = new RijndaelManaged())
                {
                    symmetricKey.BlockSize = 256;
                    symmetricKey.Mode = CipherMode.CBC;
                    symmetricKey.Padding = PaddingMode.PKCS7;
                    using (var decryptor = symmetricKey.CreateDecryptor(keyBytes,
ivStringBytes))
                    {
                        using (var memoryStream = new MemoryStream(cipherTextBytes))
                        {
                            using (var cryptoStream = new CryptoStream(memoryStream,
decryptor, CryptoStreamMode.Read))
                            {
                                var plainTextBytes = new byte[cipherTextBytes.Length];
                                var decryptedByteCount =
cryptoStream.Read(plainTextBytes, 0, plainTextBytes.Length);
                                memoryStream.Close();
                                cryptoStream.Close();
                                return Encoding.UTF8.GetString(plainTextBytes, 0,
decryptedByteCount);
                            }
                        }
                    }
                }
            }
        }

        private static byte[] generate256BitsOfRandomEntropy()
        {
            var randomBytes = new byte[32];
            using (var rngCsp = new RNGCryptoServiceProvider())
            {

                rngCsp.GetBytes(randomBytes);
            }
            return randomBytes;
        }
    }
```

SessionManager.cs

```
public static class SessionManager {
        private static long userID = 0;
        private static int sessionID = 0;

        public static long UserID { get => userID; set => userID = value; }
        public static int SessionID { get => sessionID; set => sessionID = value; }
    }
```

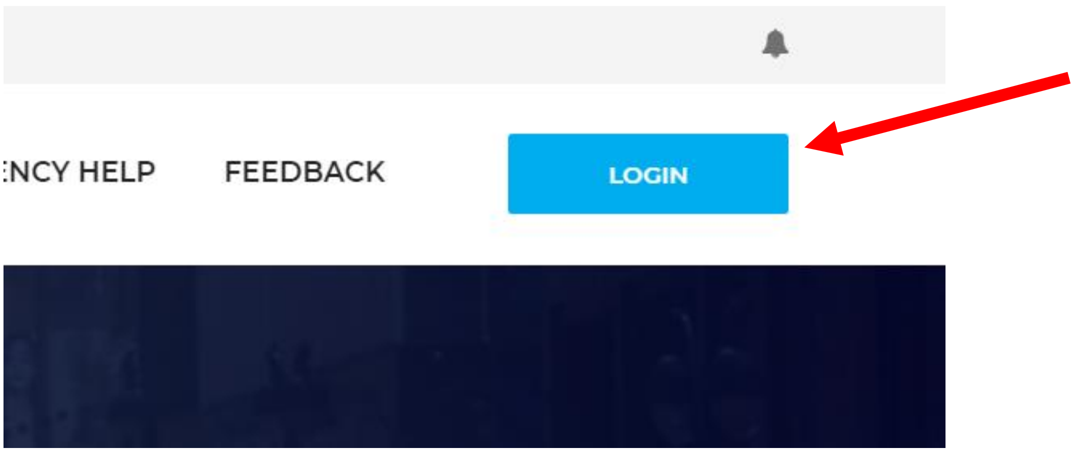
Utilities.cs

```
public static class Utilities {
        // Takes TexBoxes and checks if they're empty
        public static bool isTextBoxEmpty(List<TextBox> textBoxes) {
            foreach (TextBox txt in textBoxes) {
                if (string.IsNullOrEmpty(txt.Text)) {
                    return true;
                }
            }
            return false;
        }

        // Checks if any textBox in the view is empty
        public static bool isTextBoxEmpty(View view) {
            foreach (TextBox txt in view.Controls.OfType<TextBox>()) {
                if (string.IsNullOrEmpty(txt.Text)) {
                    return true;
                }
            }

            return false;
        }

        // Clears both TextBoxes and Placeholders
        public static void clearAll(View view) {
            foreach (TextBox txt in view.Controls.OfType<TextBox>()) {
                txt.Text = null;
            }

            foreach (PlaceHolder placeHolder in view.Controls.OfType<PlaceHolder>()) {
                placeHolder.Visible = false;
            }

            foreach (FileUpload fileUpload in view.Controls.OfType<FileUpload>()) {
                fileUpload.PostedFile.InputStream.Dispose();
            }
        }

        // Clears only the Textboxes
        public static void clearAllTextBoxes(View view) {
            foreach (TextBox txt in view.Controls.OfType<TextBox>()) {
                txt.Text = null;
            }
        }

        // Clears only the Placeholders
        public static void clearAllPlaceHolders(View view) {
            foreach (PlaceHolder placeHolder in view.Controls.OfType<PlaceHolder>()) {
                placeHolder.Visible = false;
            }
        }

        public static bool isImage(string fileExtension) {
            // Allow only files with .png, .jpeg or .jpg extensions to be uploaded.
            if ((fileExtension == ".png") || (fileExtension == ".jpeg") ||
(fileExtension == ".jpg")) {
                return true;
            } else {
                return false;
            }
        }

        public static string isSeen(bool isSeen)
        {
            if (isSeen)
            {
                return "read";
            }
            else
            {
                return "unread";
            }
        }

        public static string shortenDescription(string description)
        {
            if(description.Length > 30)
            {
                return description.Substring(0, 30) + "...";
            }
            else
            {
                return description;
            }
        }

        static Random r = new Random();

        public static string getRandomClass()
        {

            int num = r.Next(5);

            if (num <= 1)
            {
                return "success";
            }
            else if(num > 1 && num <= 2)
            {
                return "danger";
            }
            else if(num > 2 && num <= 3)
            {
                return "warning";
            }
            else
            {
                return "brown";
            }
        }
    }
```

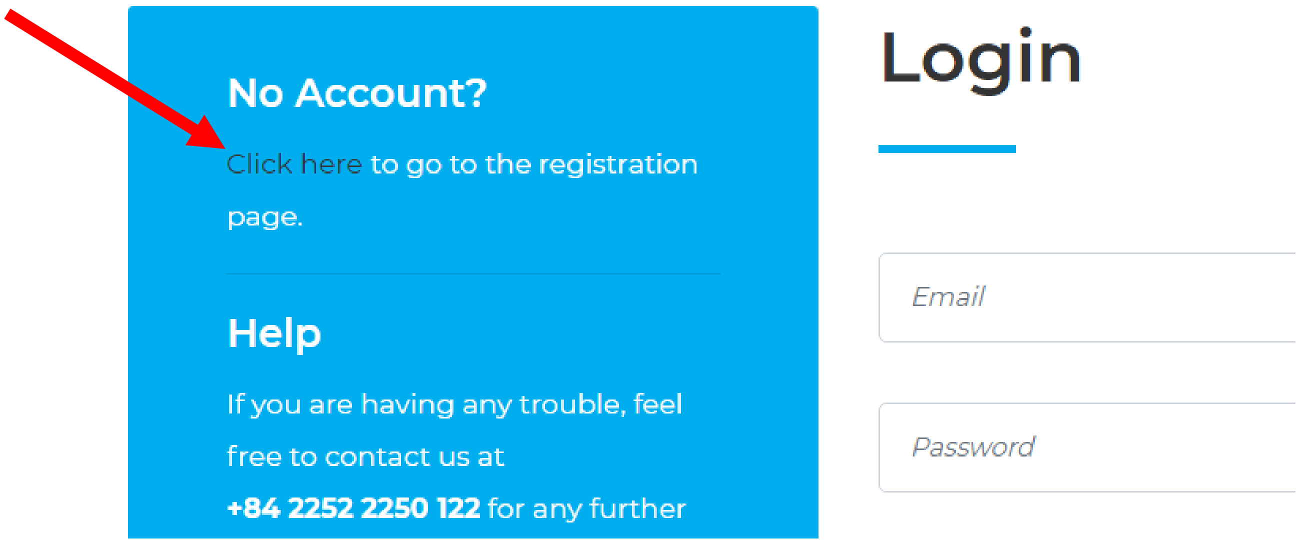
User Guide

The following contents are all that a new user needs in order to quickly grasp the concept of this web application and use most of its features effectively.

First and foremost, in order to use any feature, a user must be registered in the database. That can be done by clicking on the Login button on the top right of every page.



Upon clicking the login button, the application will take the user to the LoginRegister.aspx page, where they will initially be asked to Log in into their account. For first time users, they must register, so they should click on the register link present in the information card at the side of the login page.



No Account?

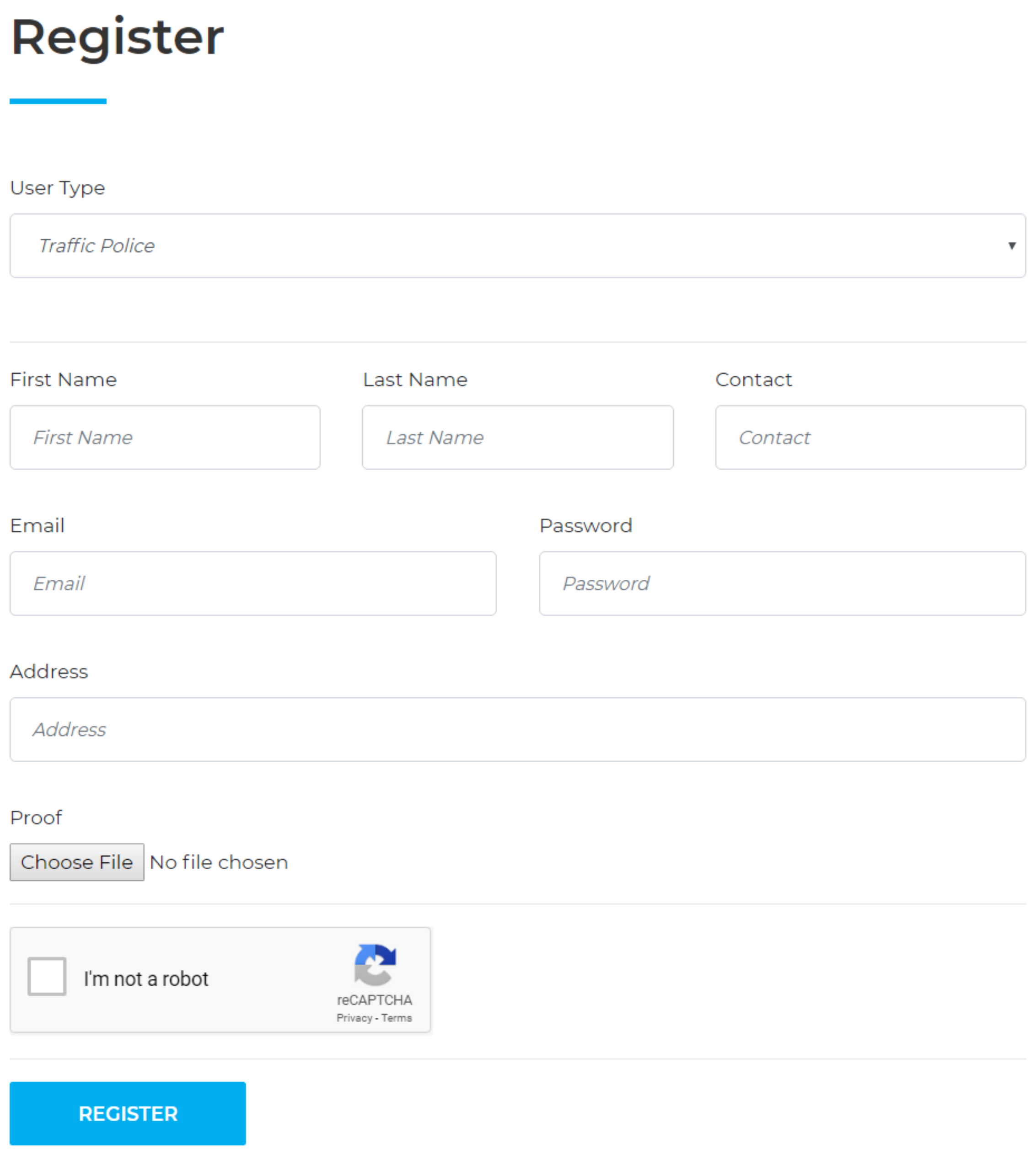
Click here to go to the registration page.

Help

If you are having any trouble, feel free to contact us at +84 2252 2250 122 for any further

Login

When the ‘click here’ link is clicked, the user is taken straight to the registration form. This is what the register form looks like:


Register

User Type

Traffic Police

First Name	Last Name	Contact
First Name	Last Name	Contact

Email	Password
Email	Password

Address

Address

Proof

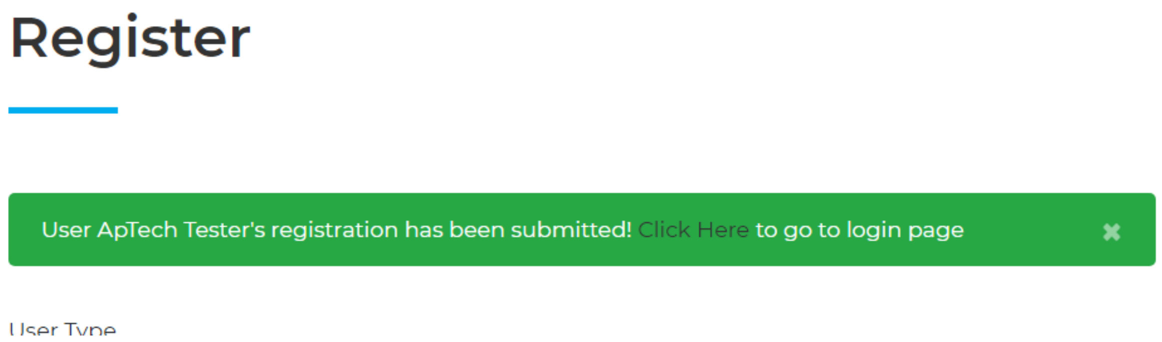
Choose File No file chosen

The new user is required to fill in all textboxes with information that is accurate to the best of their knowledge. A point to be noted is that the password has a set of special criteria of at least one Uppercase character, one lowercase character, one special character, one number, and a minimum total of eight characters. In case these criteria are not met, the user will not be able to submit the form.

The next thing to note is the Proof. A proof is an image of any document or card that holds weight in an argument in favor of the new user when they claim to be a user type of something. For example, anyone can claim to be a police officer when registering, so a proof image of their recent salary can be considered as a valid proof. Pay attention to the fact that only images of .jpg and .png format are allowed. All other formats will be rejected.

In order to avoid getting spam bots constantly registering with fake credentials, a reCAPTCHA widget has also been added to the form. Simple click on the checkbox in the widget to prove that you are human, before clicking on register. This step, like all the other ones, is compulsory in order to successfully submit an application for your registration.

Upon successful submission of the form, a confirmation message will be shown to the applying user.



Register

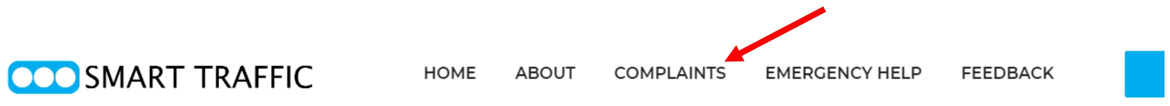
User ApTech Tester's registration has been submitted! Click Here to go to login page

User Type

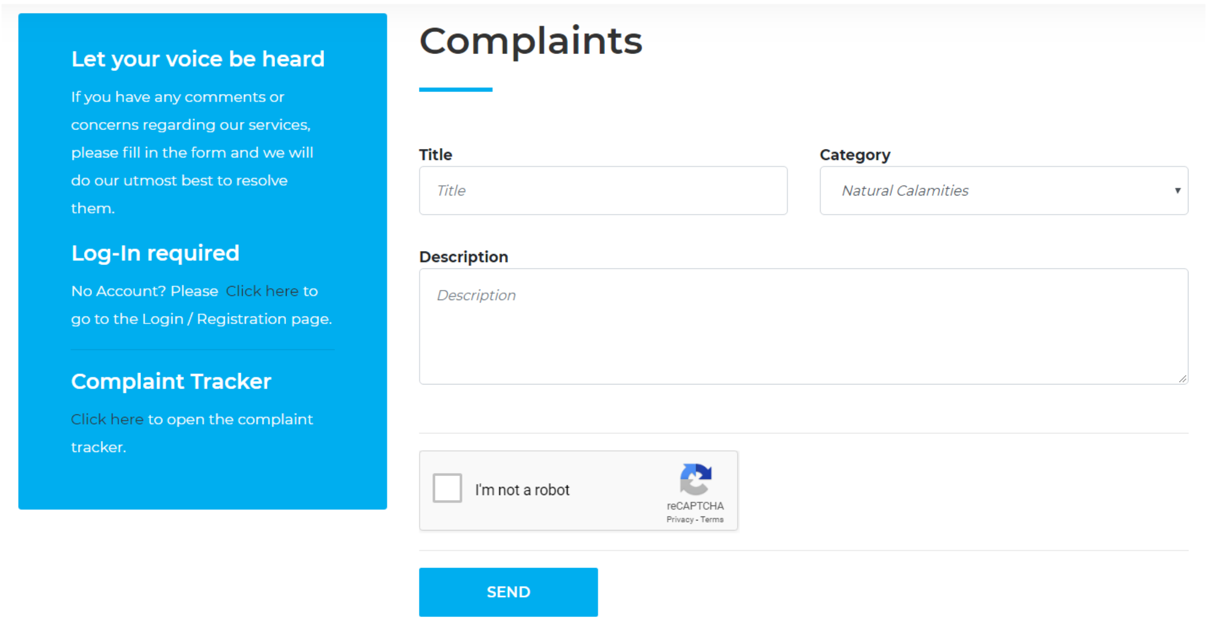
What follows is a waiting period, where the officials behind the web app will have to review the user's application, check their proof, and decide whether or not their application is accepted or rejected. Upon acceptance, a confirmation e-mail will be sent to the user to let them know.

After logging into their account, a user can utilize the various features that are offered by the Smart Traffic Management web application.

In order to file a complaint with the police department, the process is simple. First, the user must click on the 'complaints' button in the top navigation menu.



Upon clicking on the complaints button, the user will be sent straight to the complaints.aspx page of the web application.



Let your voice be heard

If you have any comments or concerns regarding our services, please fill in the form and we will do our utmost best to resolve them.

Log-In required

No Account? Please Click here to go to the Login / Registration page.

Complaint Tracker

Click here to open the complaint tracker.

Complaints

Title: Title

Category: Natural Calamities

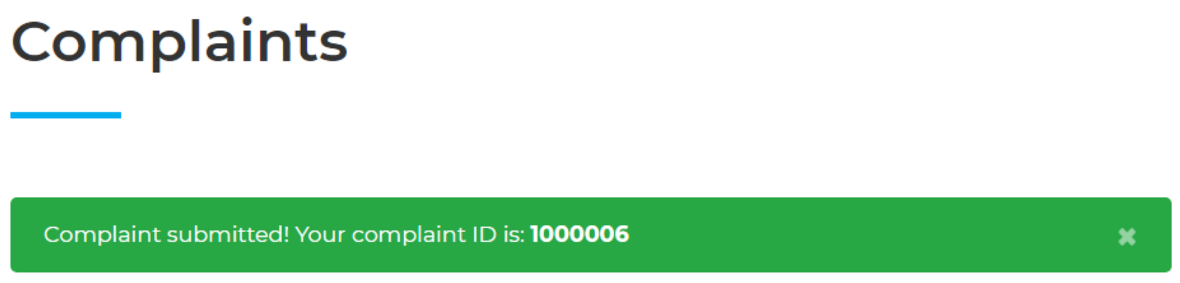
Description: Description

I'm not a robot reCAPTCHA Privacy - Terms

SEND

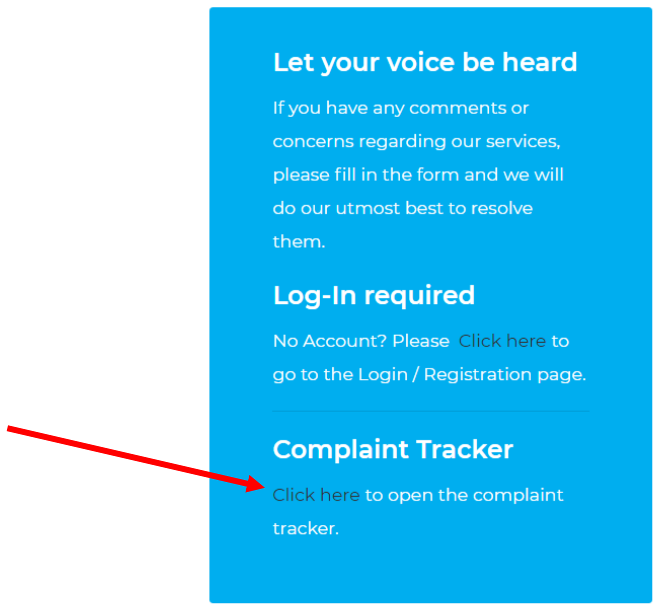
In order to file a complaint, a user needs to provide a title for their complaint, select a related category, provide a description, and prove that they are not a robot. When done, they can click on the 'send' button to submit their complaint.

Upon successful submission of the complaint, a success message will be displayed to let the user know their complaint ID.


Complaints

Complaint submitted! Your complaint ID is: **1000006**

The purpose of sharing the complaint ID with the user is for use in the complaint tracker. The complaint tracker can be easily accessed on the information side bar in the complaints page.


Let your voice be heard

If you have any comments or concerns regarding our services, please fill in the form and we will do our utmost best to resolve them.

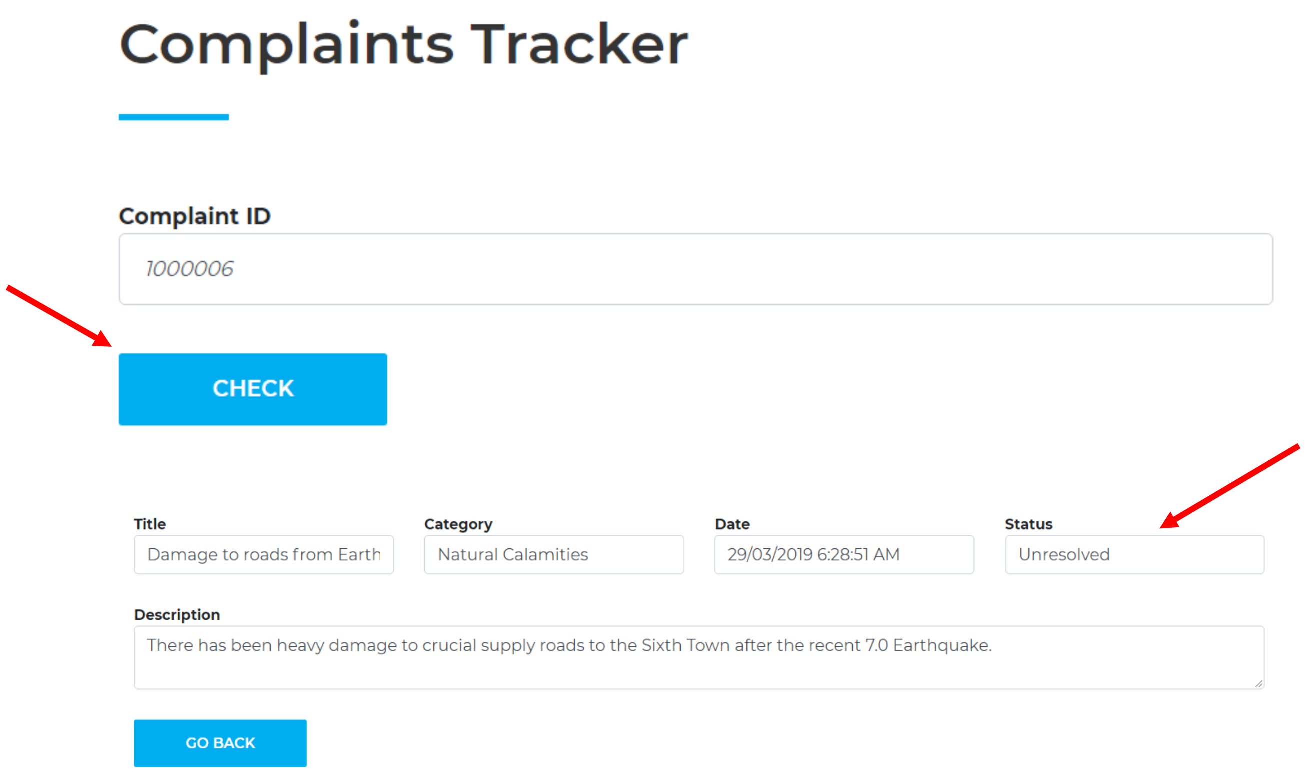
Log-In required

No Account? Please Click here to go to the Login / Registration page.

Complaint Tracker

Click here to open the complaint tracker.

When the pointed link is clicked, the user is taken to the complaint tracker. There, they can simply input their complaint ID into the single textbox, and simply press the 'check' button to search for the details of their complaint.



Complaints Tracker

Complaint ID

1000006

CHECK

Title	Category	Date	Status
Damage to roads from Earth	Natural Calamities	29/03/2019 6:28:51 AM	Unresolved

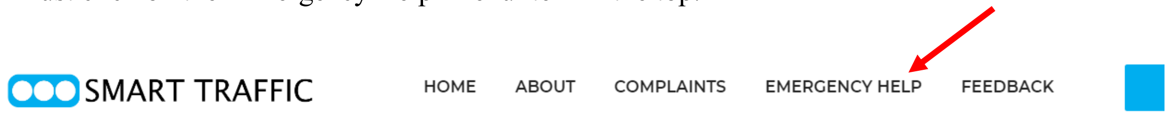
Description

There has been heavy damage to crucial supply roads to the Sixth Town after the recent 7.0 Earthquake.

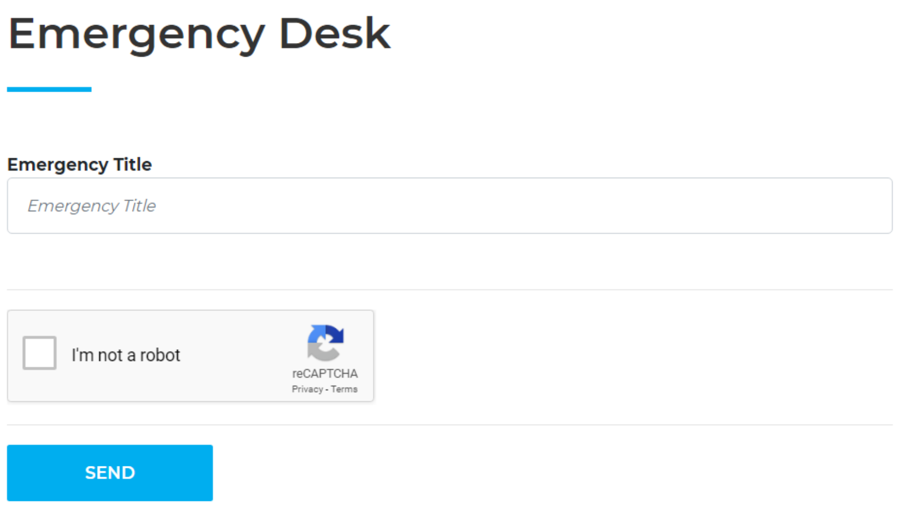
GO BACK

As seen above, a user can easily check up on the status of their complaint.

The next feature of this application is the Emergency Help request. In order to access this feature, the user must click on the 'Emergency Help' menu item in the top.



The interface for the emergency form is rather straightforward in order to avoid unnecessary wastage of time in case of an emergency.

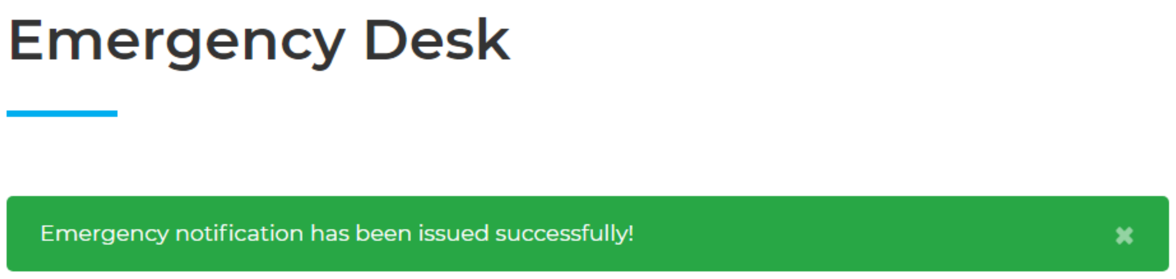


Emergency Desk

Emergency Title

Emergency Title

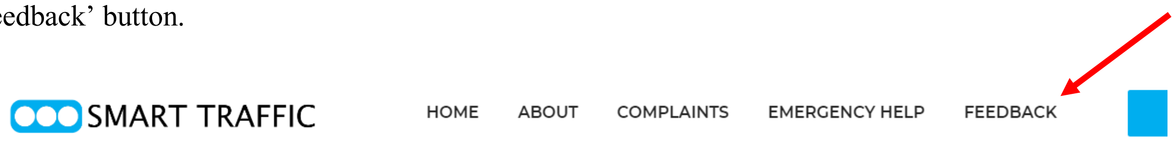
All a user needs to do is enter the title of their emergency, prove they are not a robot by using the reCAPTCHA, and click the ‘send’ button. Upon successful submission of the request, a success message will be displayed to the user as confirmation.


Emergency Desk

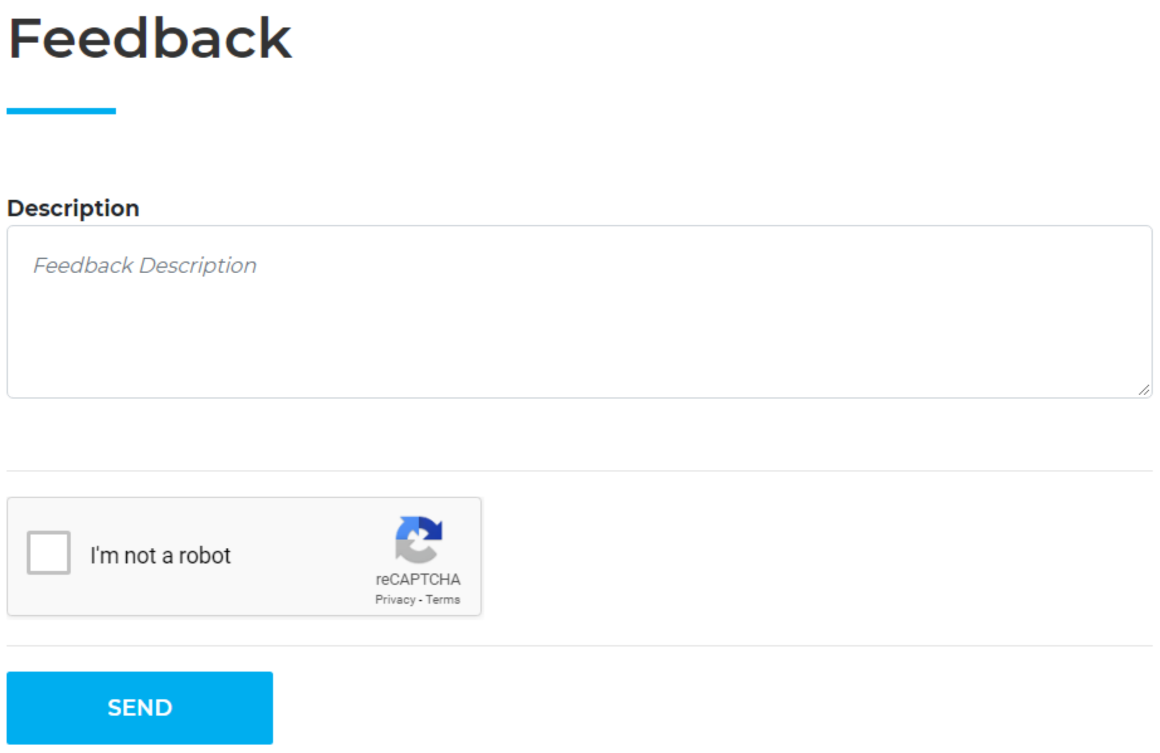
Emergency notification has been issued successfully!

The address, phone number, name, and current location of the user in need will all be automatically taken from their registered account and sent to officers so the appropriate help can be sent over as quick as possible.

The next feature of this application is Feedback. It can be accessed via the top menu by clicking on the ‘Feedback’ button.



The Feedback form is also quite simple, and doesn’t require much input from the user.


Feedback

Description

Feedback Description

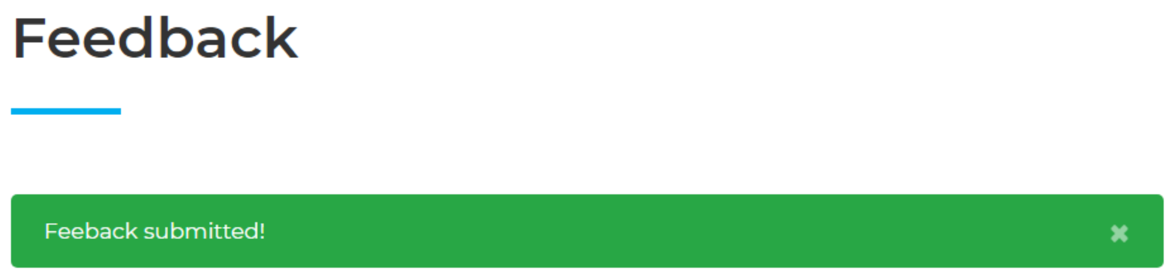
I'm not a robot

reCAPTCHA
Privacy - Terms

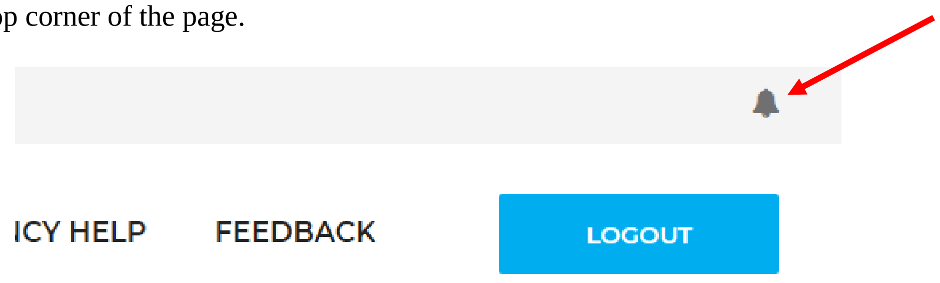
SEND

All a user needs to do is put in the description of their feedback, prove that they are not a robot by clicking on the reCAPTCHA, and clicking on the ‘Send’ button.

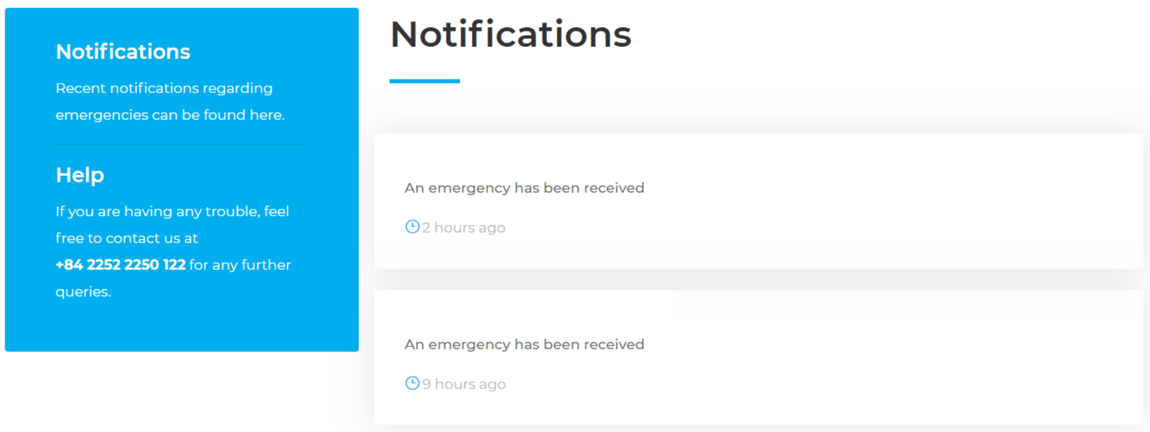
Upon successful submission of the feedback, a success message will be displayed to the user.



The next feature of this application is notifications. It can be accessed by clicking on the little bell icon on the upper top corner of the page.



Upon clicking on the notification bell icon, the user will be taken straight to the notifications page, where they can view recent notifications that have been issued by the traffic police.



Notifications

Recent notifications regarding emergencies can be found here.

Help

If you are having any trouble, feel free to contact us at **+84 2252 2250 122** for any further queries.

Notifications

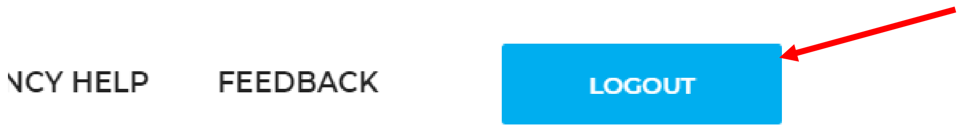
An emergency has been received

2 hours ago

An emergency has been received

9 hours ago

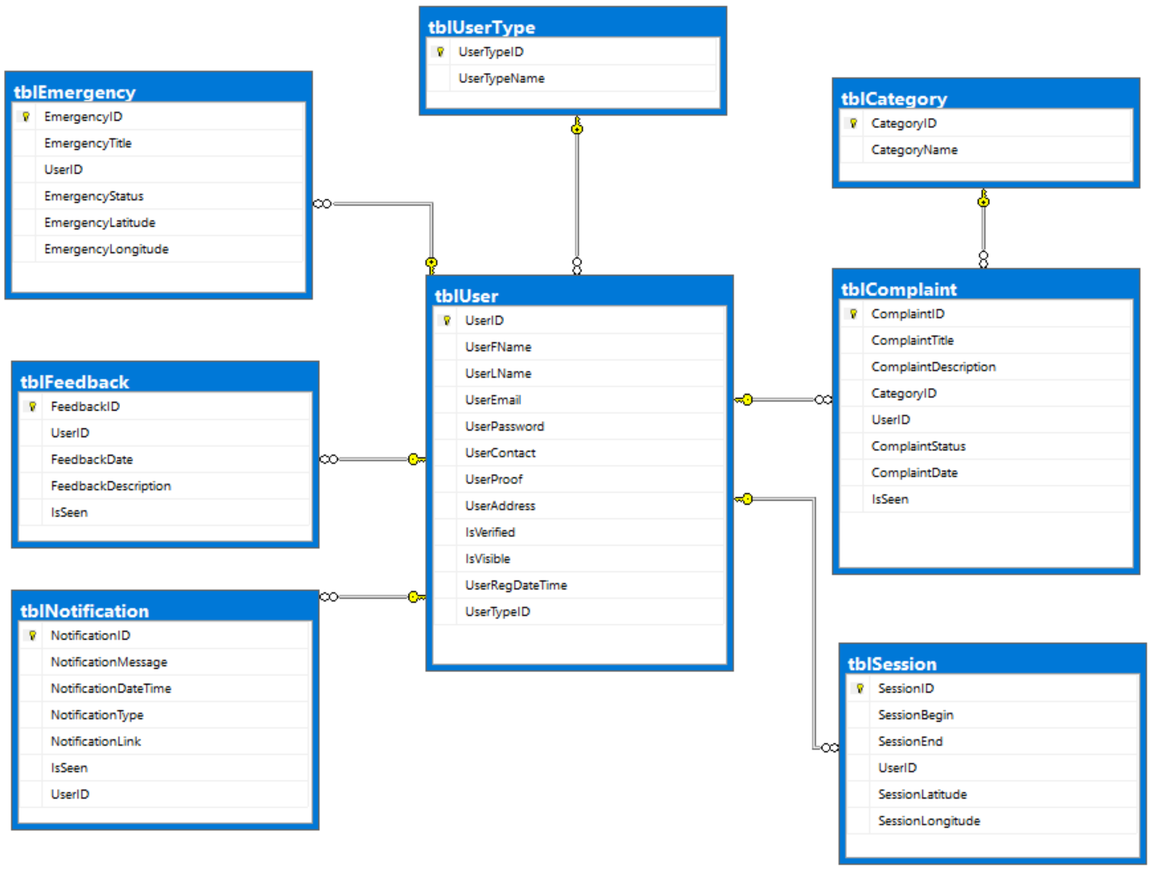
The last feature of the application is Logout. A user can easily log out of their account when they are done using the application in order to avoid letting any unauthorized person gain access to their account.



In order to view a video walkthrough, follow this link: https://youtu.be/S7ueE7mUvj0

Developer Guide

Database (SQL Server 2017)



The name of the database is SmartTrafficDB and is built upon the second layer of normalization. The primary table that this database revolves around is tblUser.

tblUser

Stores all personal data of the user. UserTypeID is selected by choosing through a DropDownList populated by entities in tblUserType. UserPassword stores the encrypted password of the user, using the user’s email as salt bytes. IsVerified by default is False and IsSeen is True; these values stay static until an official reviews the request.

tblUserType

Contains the four user types required for the web form: Official, Police Officer, Vehicle Owner and Pedestrian.

tblEmergency

Stores the emergency details reported by a user. EmergencyLongitude and EmergencyLatitude are used to pinpoint the location of the user on a map.

tblFeedback

Stores the feedback details of a user.

tblComplaint

Stores the feedback details of a user as well as allows the admin to change its status.

tblNotification

Used to store and transmit notifications from front-end to back-end. Notification type is used by the NotificationManager class to show the appropriate icon based off of it.

tblSession

Stores login and logout times of all users. SessionLongitude and SessionLatitude are refreshed every 10 seconds.

File Hierarchy

- All C# Classes are stored in the root folder.
- All ASP.NET webpages are stored in the Forms folder in either Public or Admin.
- ASP.NET webpages and their Master Page is stored in the same folder.
- All external images, CSS and JavaScript files required for the Master Page are stored in the same folder. Internal and CSS and JavaScript is used for ASP.NET webpages

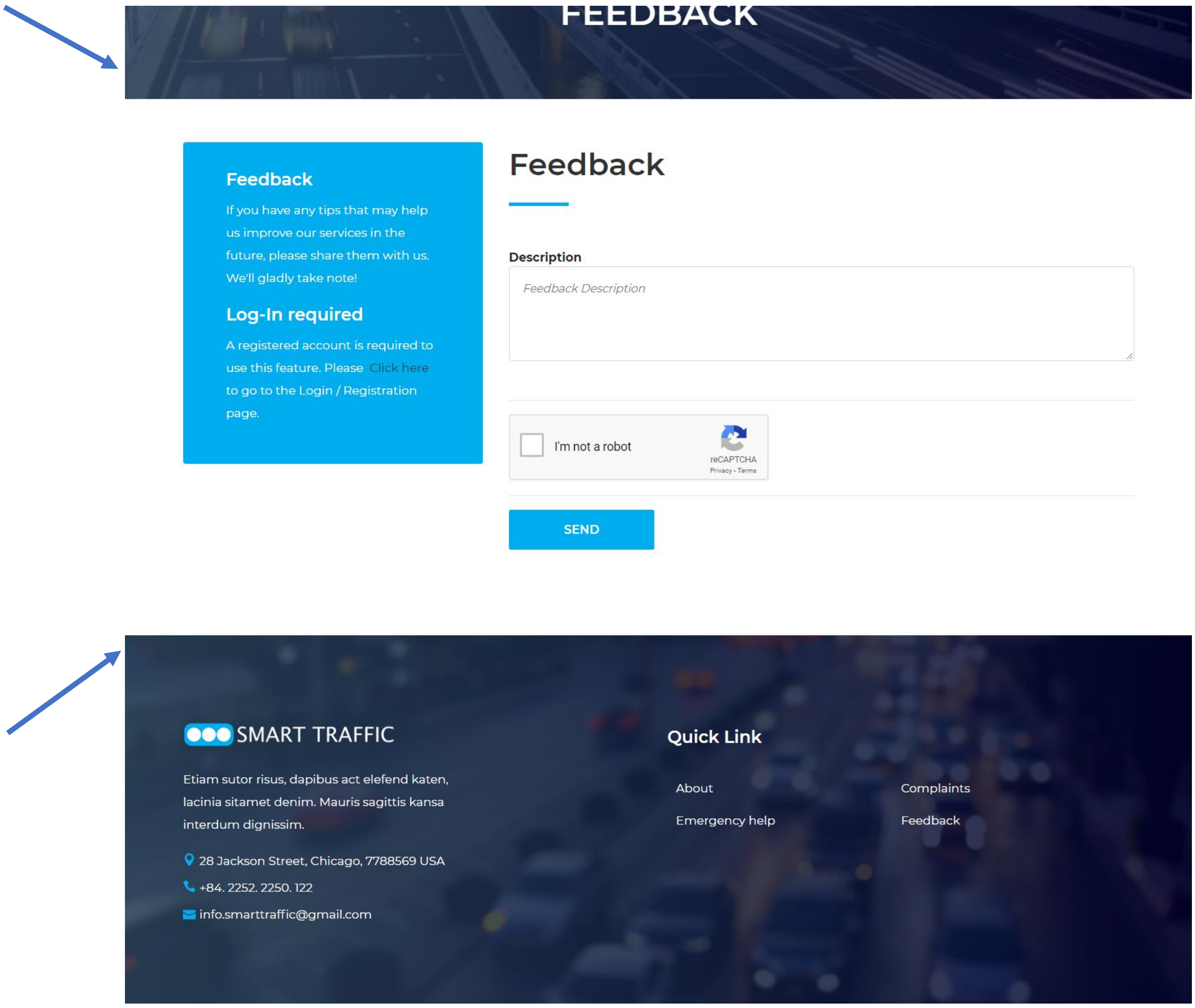
SmartTraffic

- Forms
    - Admin
        - Admin.Master
        - Alerts.aspx
        - Complaints.aspx
        - Emergencies.aspx
        - Feedback.aspx
        - Index.aspx
        - RejectedUsers.aspx
        - Reports.aspx
        - Requests.aspx
        - SessionHistory.aspx
    - Proof
    - Database.dbml
    - Public
        - About.aspx
        - Complaints.aspx
        - Emergency.aspx
        - Feedback.aspx
        - Index.aspx
        - LoginRegister.aspx
        - Notifications.aspx
        - Public.Master
- ControlManager.cs
- EmailManager.cs
- LocationManager.cs
- NotificationManager.cs

- ReCaptcha.cs
- SecurityManager.cs
- SessionManager.cs
- Utilities.cs
- Web.config

Front-end (HTML, CSS, JavaScript)

The front-end follows a generally common pattern in almost all pages. Every page contains a Header and footer, and almost all contain a banner image that mirrors the footer's look and feel, in order to give a rather 'sandwiched' appearance to the contents of the page.


FEEDBACK

Feedback

If you have any tips that may help us improve our services in the future, please share them with us. We'll gladly take note!

Log-In required

A registered account is required to use this feature. Please Click here to go to the Login / Registration page.

Feedback

Description

Feedback Description

I'm not a robot

SEND

SMART TRAFFIC

Etiam sutor risus, dapibus act elefend katen, lacinia sitamet denim. Mauris sagittis kansa interdum dignissim.

28 Jackson Street, Chicago, 7788569 USA

+84. 2252. 2250. 122

info.smarttraffic@gmail.com

Quick Link

About

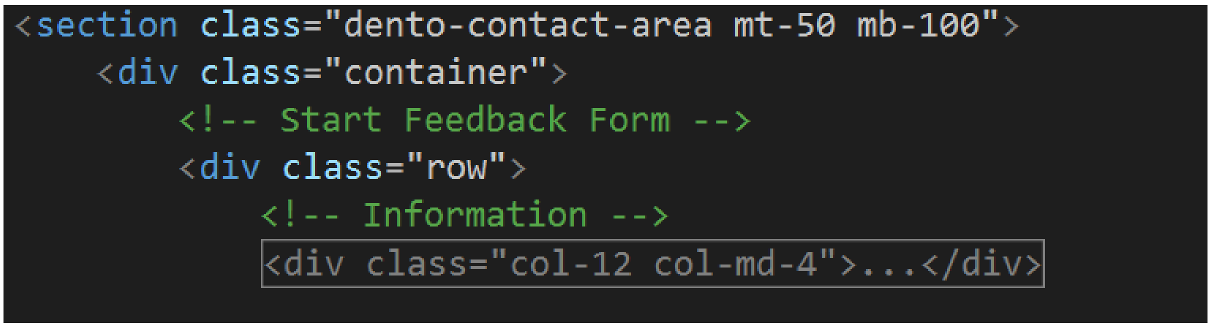
Complaints

Emergency help

Feedback

The banner follows a common format, using divs with special classes attached to them that result in the desired look and feel of the banner. It uses the breadcumb-area, bg-img, and bg-gradient-overlay CSS classes to achieve this. It also contains within it a title for the banner to let the user instantly know where they are in the web application.

The template used in the project follows the approach of having its main contents sandwiched inside a section tag, followed by a div with a container class that is appropriately named 'container', followed again by using a div with the 'row' class in every row before placing content in it. For example:



```
<section class="dento-contact-area mt-50 mb-100">
    <div class="container">
        <!-- Start Feedback Form -->
        <div class="row">
            <!-- Information -->
            <div class="col-12 col-md-4">...</div>
```

As shown here, a section tag, followed by a container, which is then followed by a row is required for the appropriate look and spacing to be achieved in this template.

When there is need of content on a new line, the previous div with the row class has to end, and another new div with yet another row class in it must be used for optimal spacing.

In many of the pages in the front end, a blue side-menu is visible which provides helpful tips and aid navigation for the user so they can get the most out of this web application. This is the ‘information’ side panel which uses a div with the ‘contact-information’ class after being sandwiched inside another div with the bootstrap class of col-12, col-md-4, in order to ensure that the information side panel does not obstruct the experience of glitch out of view in every platform the web application is used in.

Installation Guide

Requirements

- Visual Studio 2015 or higher
- Microsoft SQL Server 2010 or higher
- .NET Framework 4.0 or higher
- A modern web browser

Procedure

1. Open Microsoft SQL Server and execute script.sql
    a. If the script does not execute, manually create a database named ‘SmarTrafficDB’ and remove the first 7 lines of the script.
    b. Execute the script.
2. Open SmartTraffic.sln in Visual Studio
3. Go to Server Explorer and create a Data Connection with SmartTrafficDB
4. Install NuGet packages:
    a. EntityFramework v6.2.0 or higher
    b. Microsoft.CodeDom.Providers.DotNetCompilerPlatform v2.0.1 or higher
    c. NewtonSoft.Json v12.0.1 or higher
5. Open any .aspx file SmartTraffic/Forms and Run.