

IEEE STANDARD 1016: Software Design Specification

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behavior models, and other supporting requirement information.

The Software Design Specification Outline

1. Introduction

Police and other law enforcement agencies are dependent on highly professional and reliable crime scene technicians and investigators to uncover evidence that will prove justice and rightful information in a court of law. This system is a step to making task of citizens easier by registering their issues online through a fast working portal / server and bridging a gap b/w many people and police. This system also facilitates the police and investigators as it is a faster way for them to receive the details of various crimes taking place and saves time and speeds up the process as compared to the old register entry system. The system is completely online based platform. The idea behind having an online Crime Management system is that it'll help reporting crimes easy and help the citizen of the country as well as the police department to easily handle and manage it. This application will support citizen to file a complaint through website which will be time saving, fast, efficient and problem solving. All around country, The Crime record management application will be implemented and maintained which will again concentrate on managing complaint, prevention of crime by interconnection police information system to different police station in the country. With the usage of this information, handling of variety of criminal cases will be easy, time saving and solving cases will be much faster as each and every record will be purely accessible to the designated Anti-Crime Board. This system can handle number of users connected to the server without any glitch and error if configured correctly. Each user is authenticated and then is given an option to lodge a complaint/ first information report. Criminals; Any Citizen can report an FIR online, missing citizen search, secure registration and profile management facilities for detectives and security agencies

1.1 Purpose of this document

The purpose of this document is to discuss the details on how the software requirements should be implemented as well as it gives the programmers a blue print or a guideline to be followed. Once this document is approved by the client's team and the end developers, it serves as the reference point for the changes (if any) in the development cycle of the project.

This design document helps us to understand and evaluate if all the necessary requirements have been fulfilled. This document is more closely attached to the

development team rather than the end User or general public. In addition to this it also provides the platform to discuss the design elements which do not align with the idea of the project.

The software design document serves as a training manual for any new project members. In addition, the software design document may be used as a training manual throughout the life of the system.

This also Documents the details of the system architecture. During development phase of the this document the system architecture is finalized and inconsistencies (if any) should be resolved.

1.2 Scope of the development project

The system will be a Reporting application that will be used for automating the manual process of register different case in police Department. This system will manage the details of all the crime who register themselves in police department. The system provides an interface to store the crime details or allows the user to search and view records. The system should have a login. The objectives of this system can broadly be listed as follows:

1. Make the Police functioning citizen friendly and more transparent by automating the functioning of Police Departments.
2. Improve delivery of citizen-centric services through effective usage of this system.
3. Provide the Investigating Officers of the Civil Police with tools, technology and Information to facilitate investigation of crime and detection of criminals.
4. Improve Police functioning in various other areas such as Law and Order, Traffic Management etc.
5. Facilitate Interaction and sharing of Information among Police Departments, Districts, State/UT headquarters and other Police Agencies.
6. Assist Senior Police Officers in proper administration of Police Force
7. Keep track of the progress of Cases, including in Courts
8. Reduce manual and redundant Records keeping

They are two main advantages of this proposed system:

- 1) Reducing the crime and disorder
- 2) confidentiality and anonymity issues.

Truly Unlimited! - No restriction on no. of users. You completely own whole system. Complete freedom to live hassle free of the queues and dodging the crowds that we frequently face at these type of Govt. Offices.

1.3 Definitions, acronyms, and abbreviations

This document provides the software requirements and expected behaviour of the online portal. The document provides certain statements and requirements in a bold or highlighted format to represent the significance of the same. Detailed notes along with reference to the other documents are provided wherever applicable with an

asterisk. All the assumptions that are made will be mentioned beforehand and no external assumptions are required.

1.4 References

This website has been prepared on the basis of discussion with Team members and our course in charge. We have also taken information from following website–

- 1) www.solutions24h.com
- 2) www.logicsystems.org.in
- 3) csetekquest.blogspot.com
- 4) www.microsoft.com/sql

Weblinks:

https://en.wikipedia.org/wiki/Software_requirements_specification

Books:

Software Engineering by Roger S Pressman 5th edition, Software Engineering by Pankaj Jalote 4th edition.

Asp.net with C# Edition: First Edition Author: Shyam N. Chawada

Complete Reference C# 2.0 Edition: Second Edition Author: Herbert Schildt

UX Design: -

Don't Make Me Think: A Common-Sense Approach to Web Usability

Pervasive Information Architecture: Designing Cross Channel User Experiences

1.5 Overview of document

Section 1.0 introduces the project.

Section 2.0 provides an abstract view of the system architecture, including the components, structure and relationships, and user interfaces.

Section 3.0 describes each of these components in more detail, including design and architectural decisions.

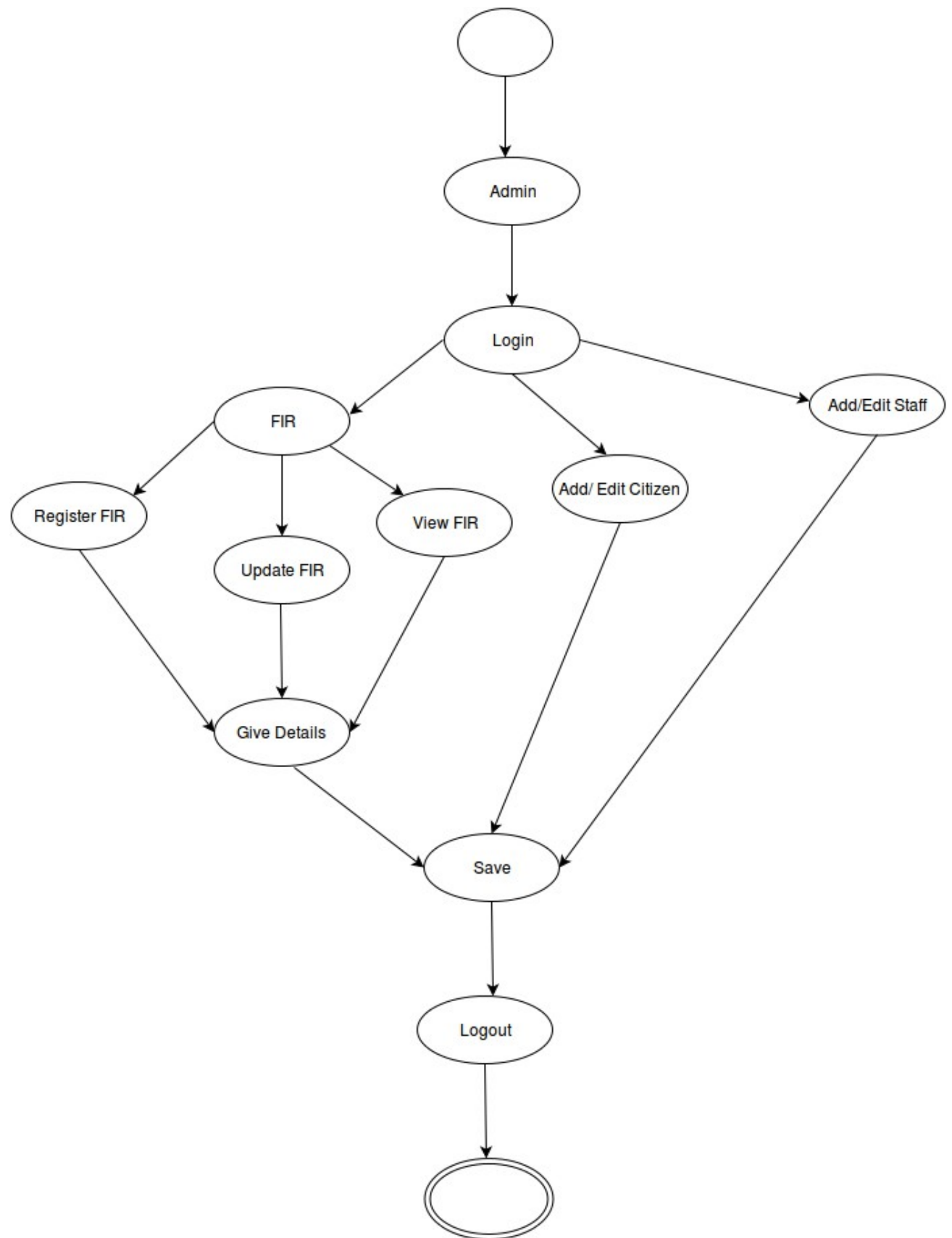
Section 4.0 explores the relationships to other products.

Section 5.0 discusses design decisions, tradeoffs, and the reasoning behind these decisions.

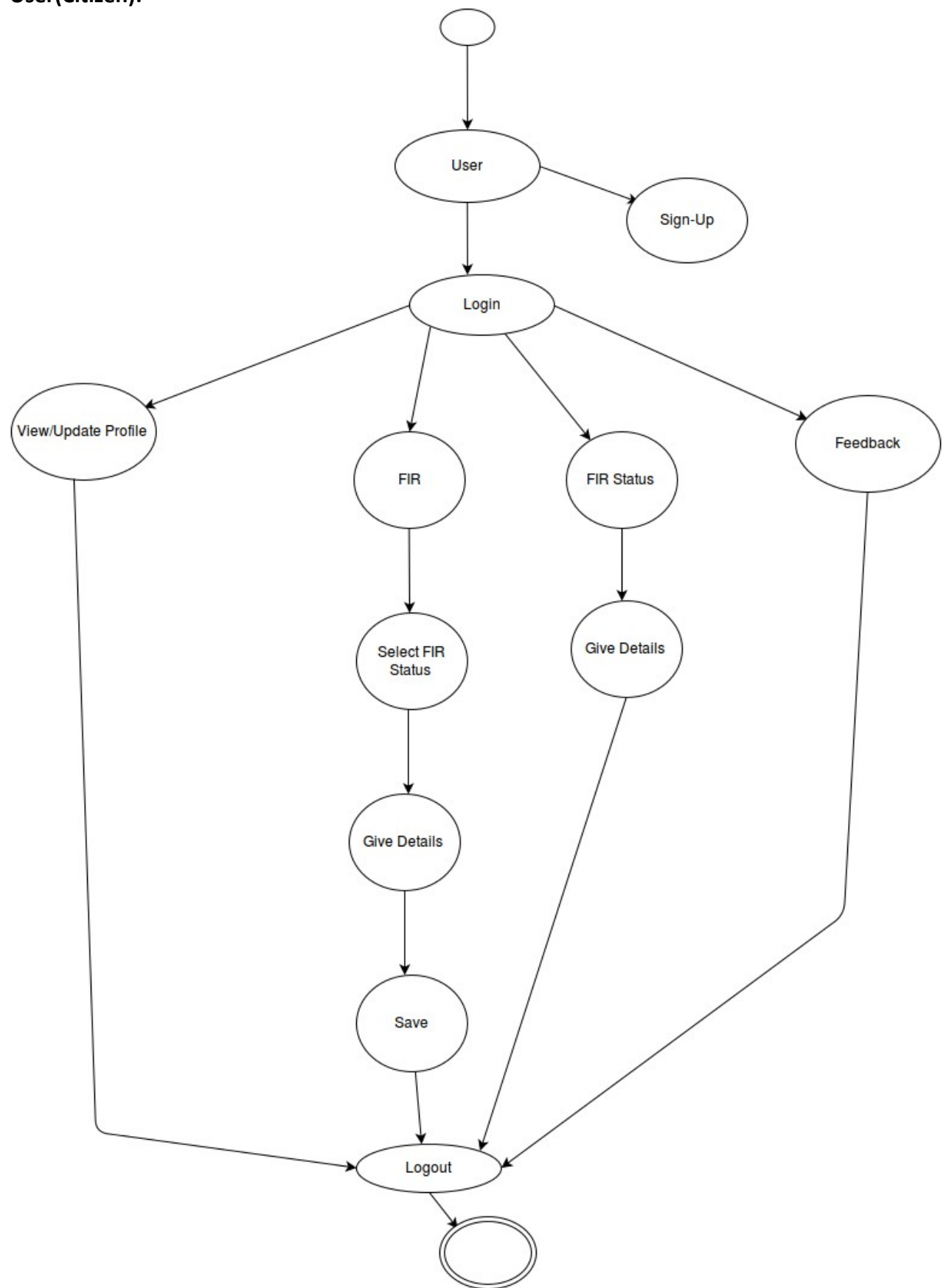
Section 6.0 is reserved for policies and tactics (pseudocode). It also discusses design patterns that can be applied.

Section 7.0 has detailed diagrams. It has both black box model and white box model.

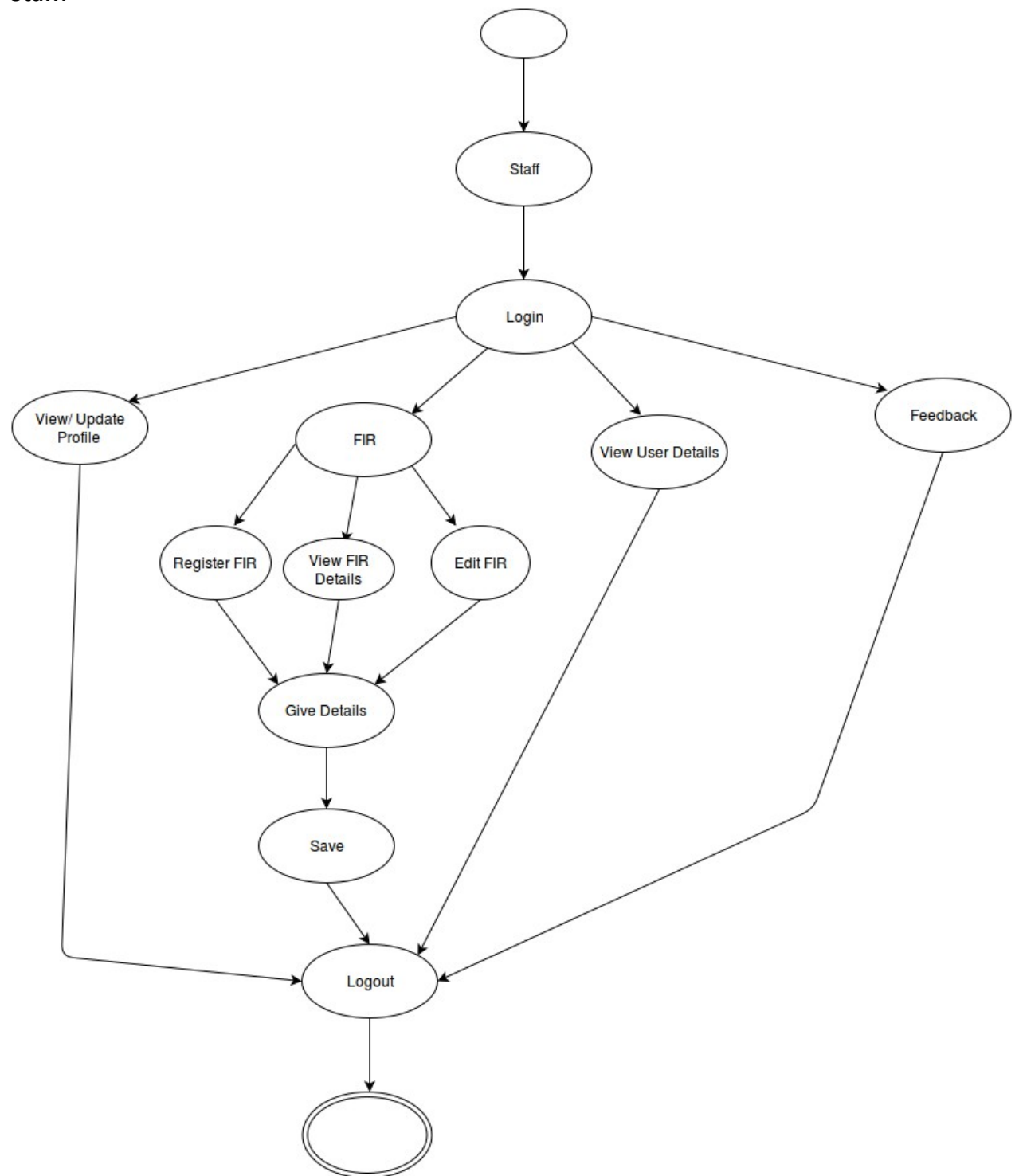
State Diagram
Admin:-



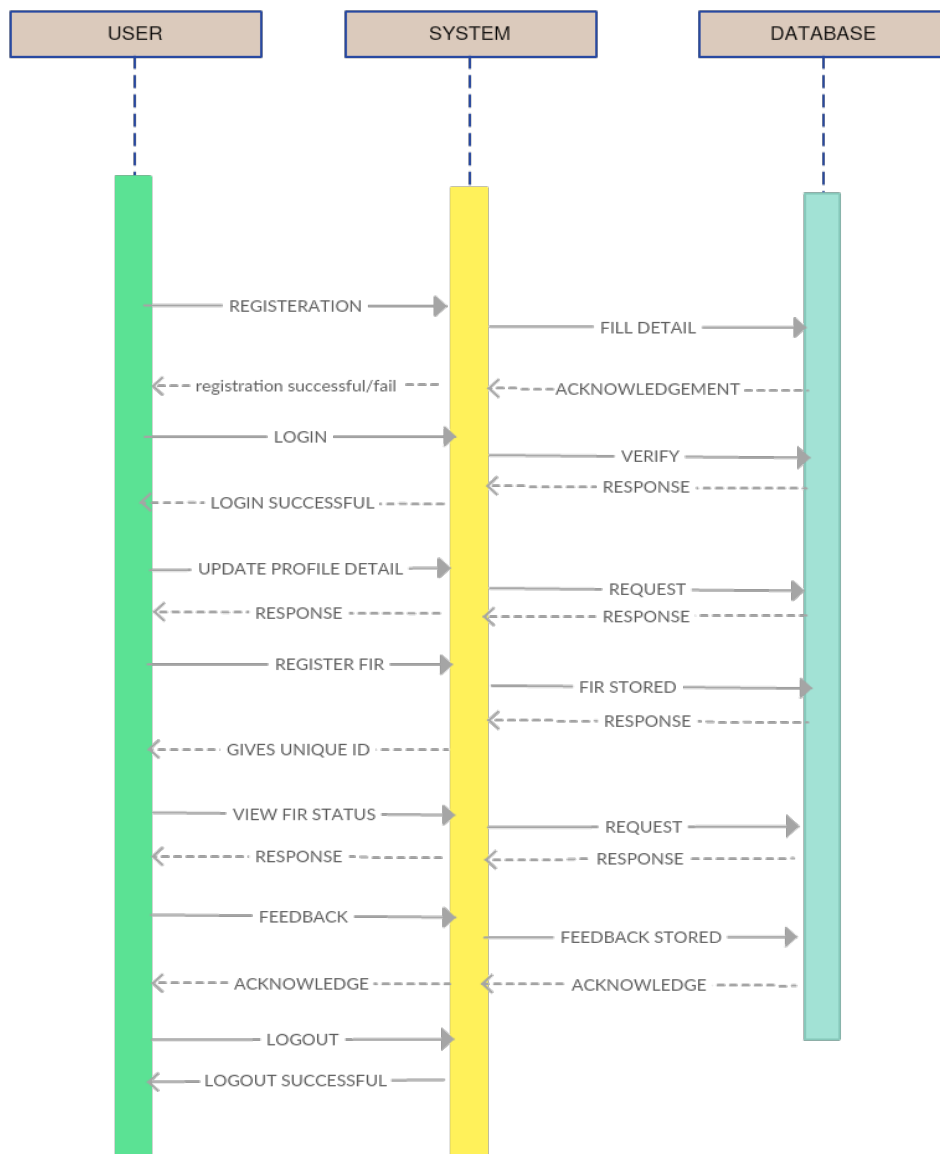
User(Citizen):-

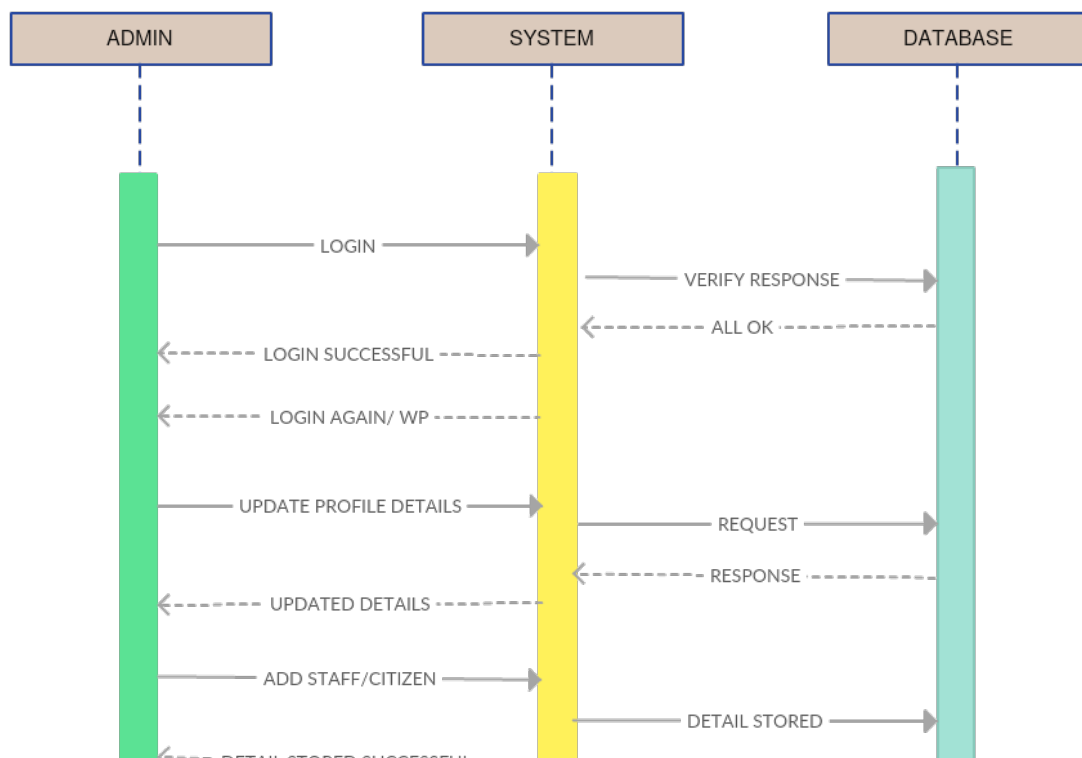
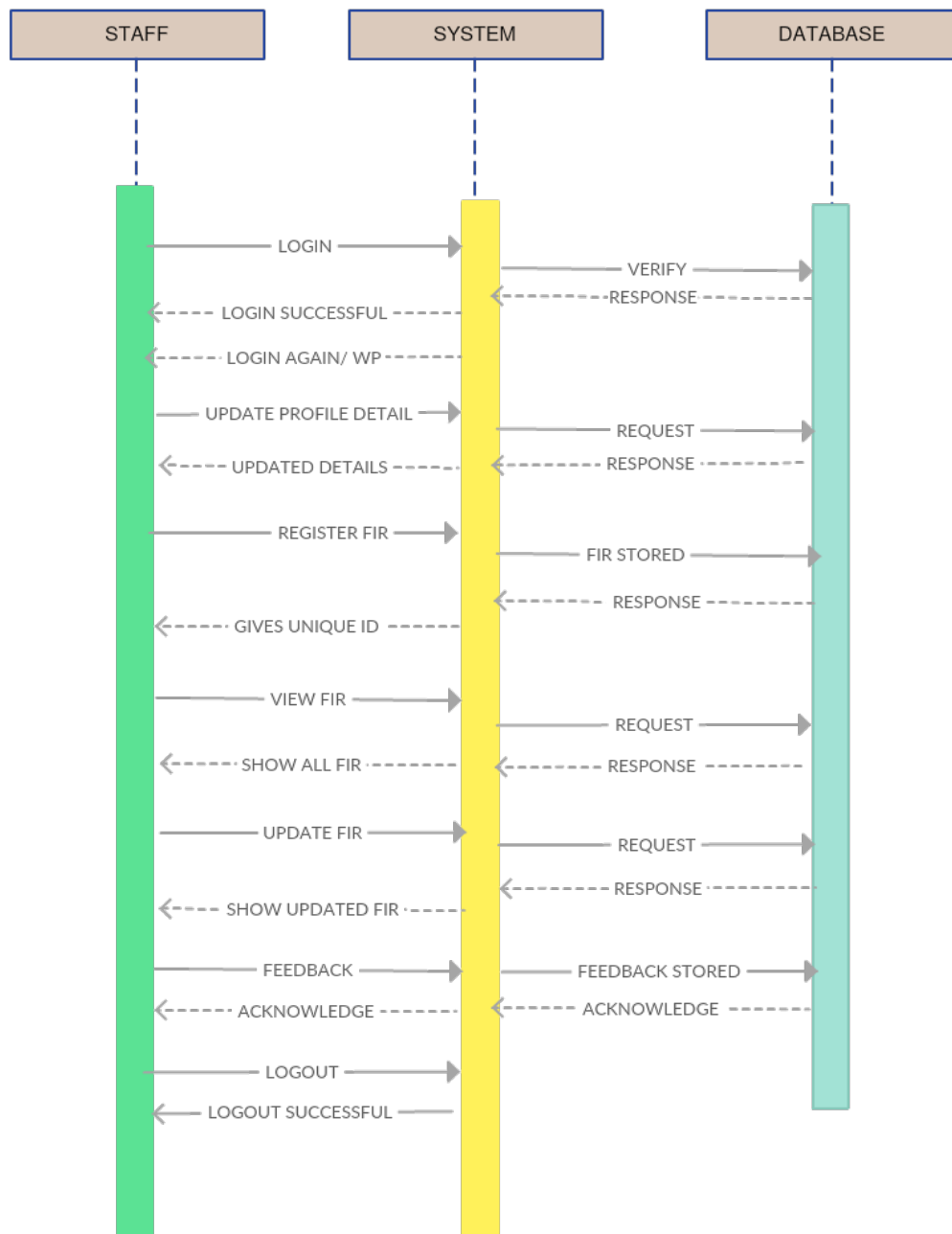


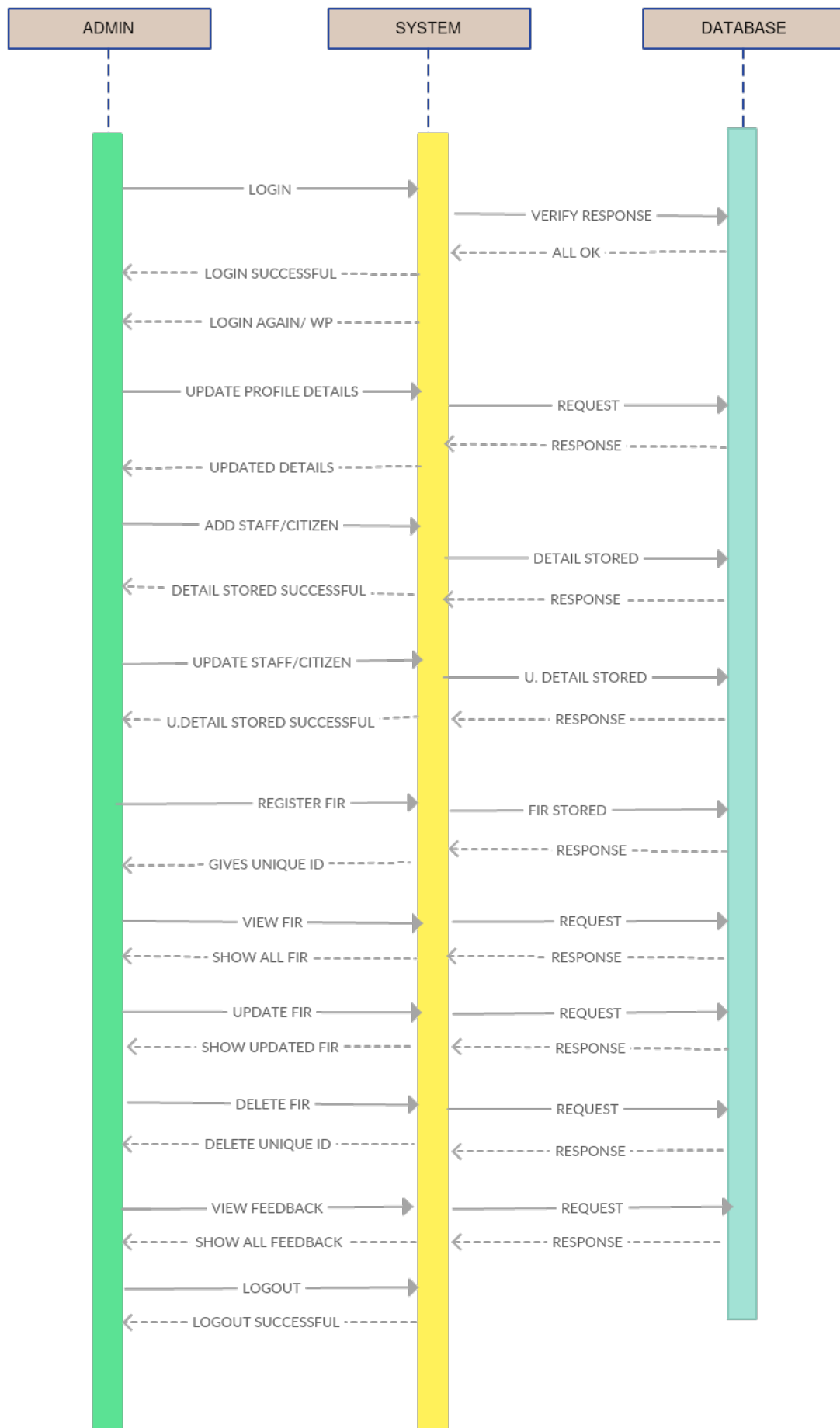
Staff:-



Sequence Diagram







2. System architecture description

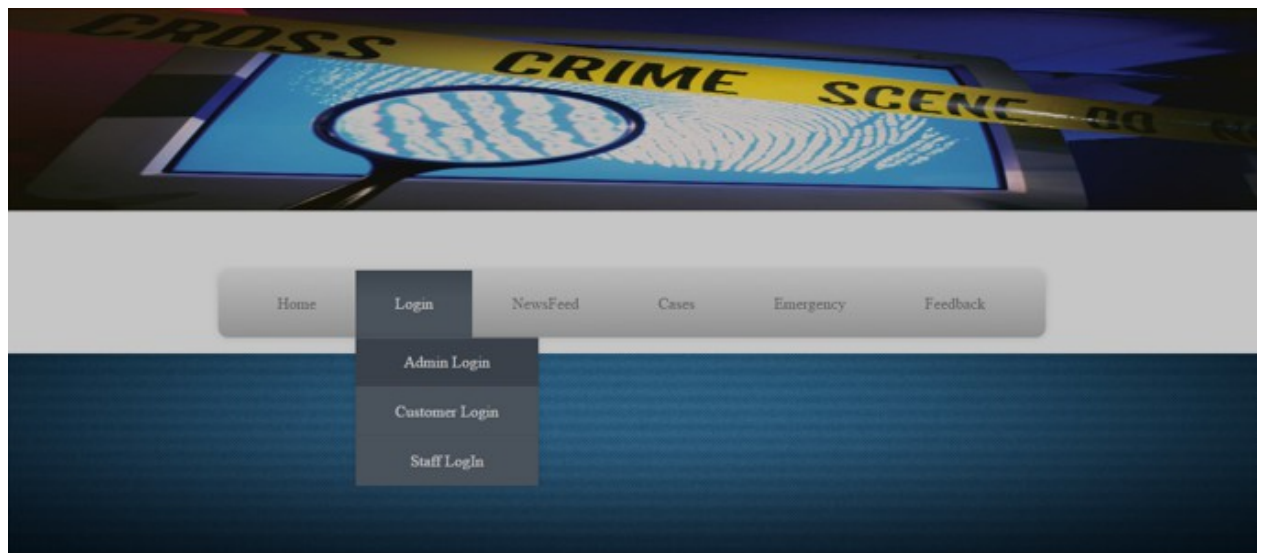
2.1 Overview of modules / components

USER INTERFACE DESIGN

This will help the user to easily interact with the application. This mainly includes the button like Home, Login, About Us, Feedback, etc. Each of them will contain fields for entering data and would also contain some information which might be useful for the user if he gets stuck anywhere in between while working on the application.

USER INTERFACE DESCRIPTION

About Main Home Page



This is the main page or the home page which consists menus like Home, Login, NewsFeed, Feedback , Cases and Emergency. On Clicking on the Home Button you will always reach this page. Below we have also mentioned the description of what our application basically is and what it does. On clicking on Login here we will get 3 options which are:-Admin Login, Customer Login and Staff Login. On clicking the Feedback button the user can give us its suggestions on what to improve and what all can be done extra.

Admin Login Page



Admin Login

User Name:

Password:

Login

On clicking on Admin Login we will get a page in which the admin of the application will get his space to enter his credentials to login (username and password).

Admin Home Page



Home Login NewsFeed Cases Emergency Feedback

Register ViewDetails Manage Settings LogOut

After Logging in the admin will get menus of Home, Login, Newsfeed, Cases, Emergency, Register, View Details, Manage Settings and LogOut.

After Clicking On Register



Register ViewDetails Manage Settigs LogOut

Register User

Register Staff

Register Case

Register FIR

Now on Clicking on the Register button the admin will get options for registering for user or staff or case or FIR.

After Clicking On Register User

HomeLoginNewsFeedCasesEmergencyFeedback

RegisterViewDetailsManage SettigsLogOut

SignUp

User Name:

Password:

Mobile:

City:

Register User

On clicking on register user, the admin can register for new user and then after the admin needs to enter the details for the new user and register it.

After Clicking On Register Staff

RegisterViewDetailsManage SettigsLogOut

SignUp

User Name:

Password:

Mobile:

City:

Register Staff

On clicking on Register Staff, the admin can register the new staff members but before that he needs to enter all the details about staff member that are username, password, mobile number and city.

After Clicking On Register FIR

RegisterViewDetailsManage SettigsLogOut

Register FIR

Registered By:

Registered Date:

Victim Name:

Victim Contact:

Victim Location:

Filed Against:

Culprit Contact:

Complaint Type: Robbery ▾

Complaint Status: Registered ▾

Register FIR

On clicking on Register FIR, the admin can register FIR for any person, and before registering the FIR he needs to fill up the details of Complaint like Registered By, Registered Date, Victim Name, Victim Contact, Victim Location, Filed Against, Culprit Contact, Complaint Type and Complaint Status.

After Clicking On View Details



On clicking on View details, the admin is asked that whose detail he/she want to view and for that he gets options that are view:- Staff Details, Citizen Details, View Cases, View FIR, View Feedback. If clicked on any of these the system will show all the details of that particular area in tabular manner for example if clicked on Staff Details then:-

Staff Details				
Citizen Details				
View Cases				
View FIR				
View Feedback				
ID	UserName	Pswrd	Mobile	City
2	Rahul	rahul	123789012	Chandigarh
3	rajeev	rajeev	147852369	merut
4	vikram	vikram	321321450	gurgaon

After Clicking On Manage Settings

After clicking on Manage account the admin will get the options for managing the citizen's account, staff account and the fir registered.

On Clicking on Citizens account, the admin gets 2 options that are Update details and Delete Account.

RegisterViewDetailsManage SettigsLogOut

Update DetailsDelete Account

On Clicking on update details, the admin will get the space to update the citizen details.

RegisterViewDetailsManage SettigsLogOut

Enter UserName:
Enter Mobile:
Enter City:
UpdateUser

And then after on clicking on UpdateUser the details gets updated.
And If clicked on Delete Account, the admin will have to enter the name of the person whose account the admin wants to delete.

RegisterViewDetailsManage SettigsLogOut

Enter Name:
Delete User

And on clicking on Delete User, the user's account gets deleted.
And If, the admin Clicks on Staff Account the also the admin gets the same 2 options as he got for the citizens.

RegisterViewDetailsManage SettigsLogOut

Update StaffDelete Staff

Now, if the Admin clicks on FIR Setting then the admin the update the FIR status.

Register
ViewDetails
Manage Settgis
LogOut

Update FIR
Select ID: 1
Complaint Status: Registered
Update

After Clicking On Customer Login

Home
Login
NewsFeed
Cases
Emergency
Feedback

[User Login](#)
User Name:
Password:
Login Register Yourself

Here the customer can login himself, and if does not have a account he can also create that and to login the customer needs to enter his username and password. After logging in the customer will get 3 options:- File FIR, Manage Account and Logout.

Home
Login
NewsFeed
Cases
Emergency
Feedback

User Profile
File FIR Manage Account LogOut

Now, if the customer goes to File FIR,

User Profile
File FIR Manage Account LogOut

Register FIR
Registered By:
Registered Date:
Victim Name:
Victim Contact:
Victim Location:
Filed Against:
Culprit Contact:
Complaint Type: Robbery
Complaint Status: Registered
Save

Then, to file the FIR the customer needs to enter all the details and he/she needs to save that, doing this will register the FIR.

Now, if the customer clicks on Manage Account, he/she will get 2 options that are update account and delete account. Clicking on Update account will help him update his own details.

User Profile

[File FIR](#) [Manage Account](#) [LogOut](#)

Enter Name:
Enter Mobile:
Enter City:

And if clicked on Delete User, then the customer needs to enter his name and click on delete button which will make his account deactivated.

User Profile

[File FIR](#) [Manage Account](#) [LogOut](#)

Enter UserName:

And if the Customer clicks on the Logout button, it will take him out from his account.

After Clicking On Staff Login

[Home](#) [Login](#) [NewsFeed](#) [Cases](#) [Emergency](#) [Feedback](#)

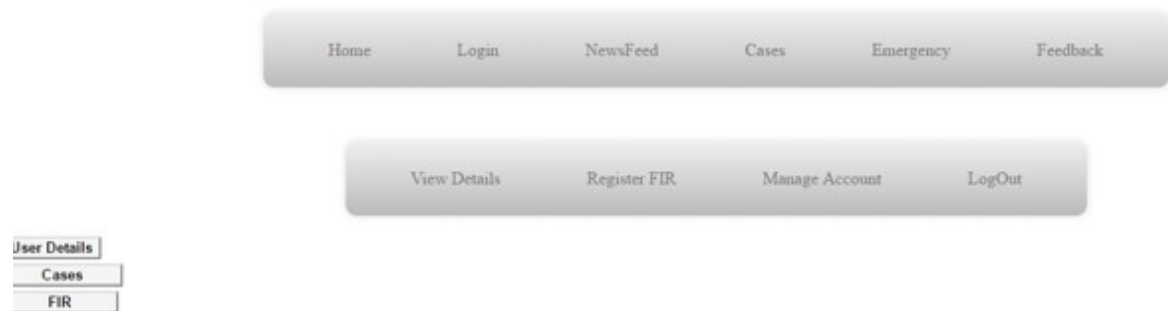
Staff Login
User Name:
Password:

The staff members needs to login to use his account and here he will get to enter the credentials to login.

After Logging in the Staff Member gets 4 options that are View Details, Register FIR, Manage Account and Logout.



If the staff member clicks on View Details then he/she gets the details about the users, all sorts of cases registered and details of all the FIR registered.



Now, if the staff member clicks on Register FIR the one needs to enter all the details to register the FIR, and then click on Register FIR to register the FIR.”

Now, if the staff member clicks on Manage Account he/she will get 2 options that are:- Update Details and Change Password.

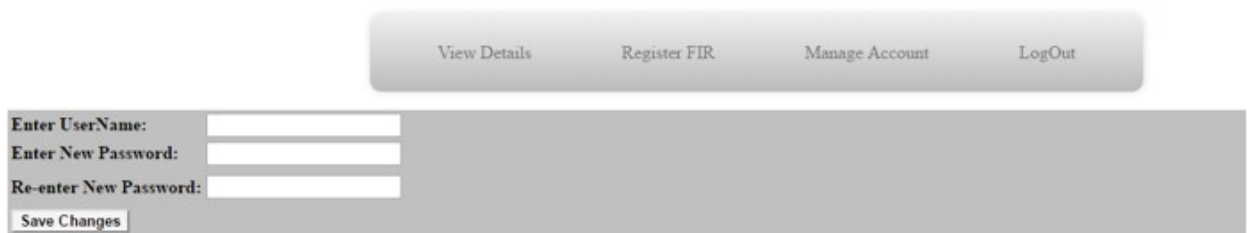
If Clicked on Update Details

The staff member needs to enter the details they wanted to update and then click on Update button.



A user profile update form. At the top, there is a horizontal bar with four buttons: "View Details", "Register FIR", "Manage Account", and "LogOut". Below this bar, the form contains three input fields labeled "Enter UserName:", "Enter Mobile:", and "Enter City:". To the right of each label is a text input box. At the bottom left of the form is an "Update" button.

If Clicked On Change Password



A change password form. At the top, there is a horizontal bar with four buttons: "View Details", "Register FIR", "Manage Account", and "LogOut". Below this bar, the form contains three input fields labeled "Enter UserName:", "Enter New Password:", and "Re-enter New Password:". To the right of each label is a text input box. At the bottom left of the form is a "Save Changes" button.

If clicked on Change Password the staff member needs to enter the user name, and he needs to enter the new password then again re-enter it for the confirmation and then click on Save Changes button for saving the password.

If the staff member clicks on Logout it would take them out of their account to the main home page.

INTERFACE DESIGN RULES

1. Efforts For The Flexibility of Application

Consistent sequences of actions are required in same kind of situations and identical terms are used in menus, home and feedback, and the consistent terms are used throughout.

For each menu in our application we have represented them with their names on the button itself for example "Admin Login", "Citizen Login", etc. The buttons belong to their particular group. And all the main menu is there on the top of the screen.

2. Use Of Shortcuts Periodically

Use of shortcuts is directly proportional to the user's interaction towards the application and also at the same time it also increases the user's speed of interacting with the application. Keys like functional keys and hidden feedback are helpful to the user.

Also for the ease of user we have grouped each and every menu in their particular section. For example: -Register FIR is a Button inside Citizen Login.

3. Feedback Section

Users are provided with a feedback option in Login Page itself, in which user can provide us with some useful information about the application like what else we can improve in our system or if there is something wrong like server is running low, they can provide us with such information. And also, whenever there is any feedback from the user we will get a notification for that.

4. Organize the entities into Cluster

Actions are mentioned to their particular group.

5. Error Handling

_Design the system in such a way that user should not face any type of error and if it occurs in any case the system should be smart enough to detect that error and should help the user with the solution of that particular error.

6. Reversibility of Actions

This feature allows the user to easily go back or to undo the actions they have done, so the user knows that the errors can be undone and so it helps in the exploration of unfamiliar actions. It may be a single user action or a complete group action. For example, :-If the admin adds a member by mistake so he can remove him and also, he ca re-add him.

7. Center Support of Control

Design the system in such a way that the users are the initiators of the actions rather to keep them only as responders. Users using the application from a long period of time assume themselves as the in charge of the system and the system should response to their actions.

8. Long-Term Processing

For Long term processing the display should be kept simple, multiple page display be consolidated, reduce window motion frequency and actions should be in sequence. Each menu is moving towards a single task and make menu in such a way that it should be quickly understandable by user.

COMPONENTS AVAILABLE

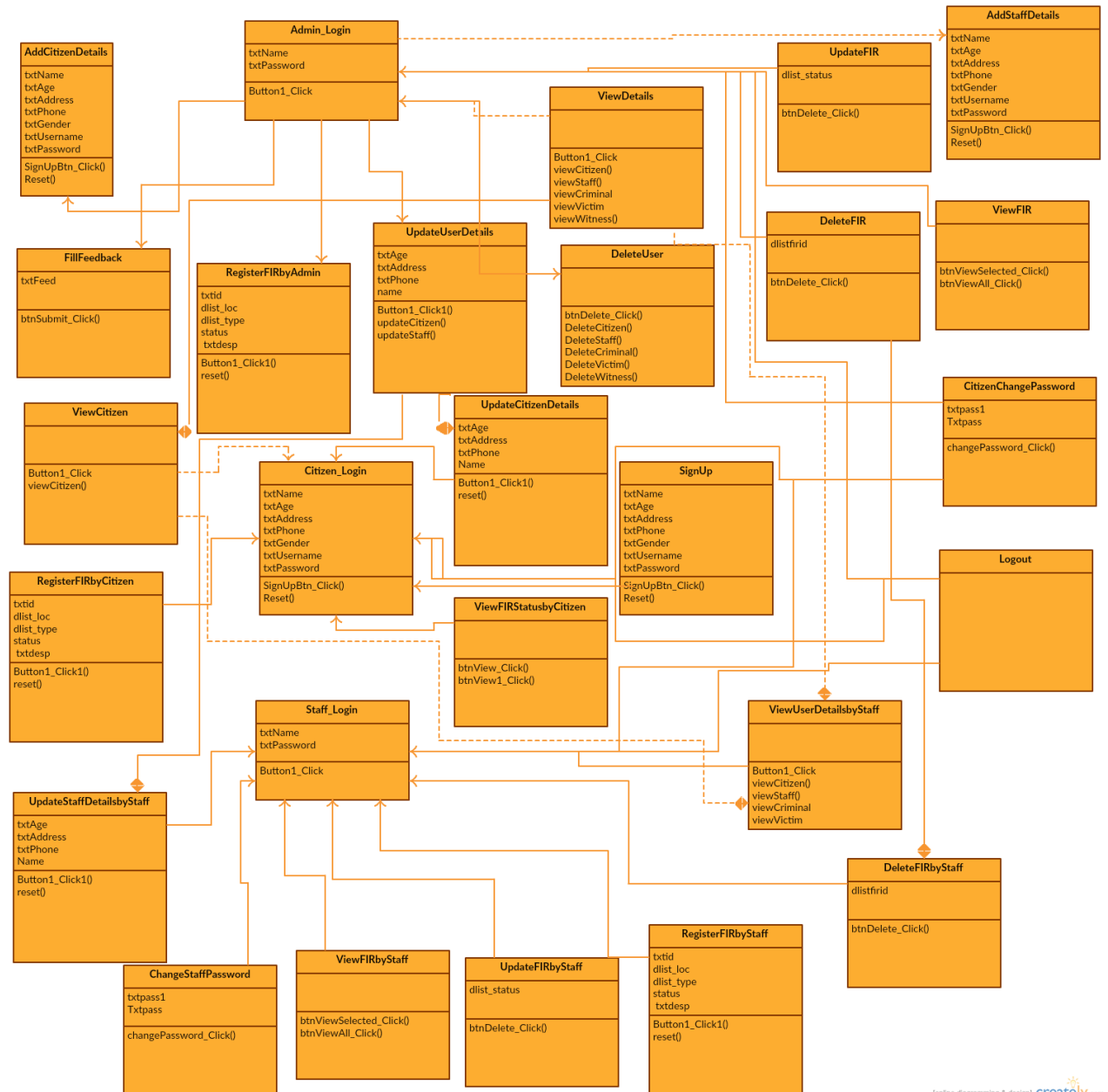
1. **View:-** It is mainly used to display the GUI Components.
All the menu's can be viewed using the Viewed Class
2. **Absolute Layout:-** It is mainly used to arrange the buttons in order.
It allows user to text in a logical manner.
3. **Button:-** It allows user to easily interact with the application, when it is pushed it creates an event
which can be handled by an EventHandler outside the button class.
4. **TextView:-** It is used to display text in a linear manner. It is mainly used to present the menu names.
5. **ListView:-** It basically displays the list of objects which can be scrolled.
It also generates an event when the selection of objects are changed.

UIDS DESCRIPTION

1. The UIDS is provided by Visual Studio (Enterprise).

2.2 Structure and relationships

Class Diagram



[online diagramming & design]  [creately.com](https://www.creately.com)

2.3 User interface issues

There are specific interface issues that have not be redressed in our User Interface Design and hence specific sections of our society may find it difficult to use the system and those are:

- Blindness so a screen cannot be read.
- Restricted sight, so that a normal display cannot be seen clearly.
- Physical impairments that restrict the use of a keyboard and/or mouse.
- Colour blindness, so that certain kinds of colour separation on a screen cannot be seen

3. Detailed description of components

3.1 Component template description

This section explains each class component in great detail:

Component Name: AddCitizenDetails	
Brief Description: A very basic Registration Process in which a Citizen has to add his/her own profile. A Citizen has to input his/her Username and password at each Login Attempt.	
Attributes	Attribute description
txtName	This will be displayed in the Profile Section. It acts as a personal identity.
txtAge	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtPhone	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtGender	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtAddress	This will be displayed in the Profile Section of each individual. It acts as a personal identity.

txtUserName	This will be the unique identity of the users. Username will be helping the users to login to the system. This will be the primary identity of the authentication.
txtPassword	This will be secret identity of the user to logon to the system. Along with the username this secret ("Password") will be needed for a successful authentication.
Methods()	Methods() Description
SignUpBtn_Click()	Button to Signup and fill in the details
reset()	This method will nullify changes in txtName, txtAge, txtPhone, txtAddress, txtUserName and txtPassword.

Component Name: AddStaffDetails	
Brief Description: A very basic Registration Process in which a Staff Official has to add his/her own profile. A Citizen has to input his/her Username and password at each Login Attempt.	
Attributes	Attribute description
txtName	This will be displayed in the Profile Section. It acts as a personal identity.
txtAge	This will be displayed in the Profile Section of each individual. It acts as a personal identity.

txtPhone	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtGender	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtAddress	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtUserName	This will be the unique identity of the users. Username will be helping the users to login to the system. This will be the primary identity of the authentication.
txtPassword	This will be secret identity of the user to logon to the system. Along with the username this secret ("Password") will be needed for a successful authentication.
Methods()	Methods() Description
SignUpBtn_Click()	Button to Signup and fill in the Staff details.
reset()	This method will nullify changes in txtName, txtAge, txtPhone, txtAddress, txtUserName and txtPassword.

Component Name: [Citizen_Login](#)

Brief Description:

An Citizen has to login to the web application using his/her Username, Password and if both the things matches then authentication is done, then they get to access whole application.

This class even redirects to another Super Class – "SignUp".

Attributes	Attribute description
txtName	This will be displayed in the Profile Section. It acts as a personal identity.
txtAge	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtPhone	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtGender	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtAddress	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtUserName	This will be the unique identity of the users. Username will be helping the users to login to the system. This will be the primary identity of the authentication.
txtPassword	This will be secret identity of the user to logon to the system. Along with the username this secret ("Password") will be needed for a successful authentication.
Methods()	Methods() Description
SignUpBtn_Click()	Button to Signup and fill in the citizen details.
reset()	This method will nullify changes in txtName, txtAge, txtPhone, txtAddress, txtUserName and txtPassword.

Component Name: Admin_Login	
Brief Description: An Admin has to login to the web application using his/her Username, Password and if both the things matches then authentication is done, then they get to access whole application.	
Attributes	Attribute description
txtName	This will be the unique identity of the Admin. His/Her name will be helping the Admin to login to the system. This will be the primary identity of the authentication.
txtPassword	This will be secret identity of the Admin to logon to the system. Along with the username this secret ("Password") will be needed for a successful authentication.
Methods()	Methods() Description
Button1_Click	It allows the Admin inputting username along with the password.

Component Name: Staff_Login	
Brief Description: A Staff Official has to login to the web application using his/her Username, Password and if both the things match then authentication is done, then they get to access whole application.	
Attributes	Attribute description
txtName	This will be the unique identity of the Admin. His/Her name will be helping the Admin to login to the system. This will be the primary identity of the authentication.

txtPassword	This will be secret identity of the Admin to logon to the system. Along with the username this secret ("Password") will be needed for a successful authentication.
Methods()	Methods() Description
Button1_Click	It allows the staff inputting username along with the password.

Component Name: AdminProfile
Brief Description: An AdminProfile is a partial class which can be used to check the Admin's Details when it gets Admin Details directly registered in the database by Police Headquarters. It functions as just displaying the whereabouts of an Admin.

Component Name: Home
Brief Description: Home is a partial class of Login which redirects the user to Home Page. (Master Page)

Component Name: StaffProfile
Brief Description: StaffProfile is a partial class which redirects the System Staff to again redirects to a HomePage.
Component Name: CitizenProfile

Brief Description:

An CitizenProile is a partial class which can be used to check the Citizen's Details when it gets Citizen Details directly registered in the database by Police Headquarters. It functions as just displaying the whereabouts of Citizen.

Component Name: [Contact_Info](#)

-*Brief Description:

An Contact_Info is a partial class which can be used to check the Citizen's, Admin's and Staff's Contact Info respectively when it gets Requisite Details directly registered in the database by Police Headquarters. It functions as just displaying the whereabouts of Required Body.

Component Name: [ViewCitizen](#)

-*Brief Description:

An ViewCitizen is a partial class which can be used to check and view their own Citizen Details which were recorded at the very start through registration. It functions as just displaying the whereabouts of Required Body.

Component Name: [ViewFeedback](#)

-*Brief Description:

An ViewCitizen is a partial class which can be used to check and view their own Citizen Details which were recorded at the very start through registration. It functions as just displaying the whereabouts and reflects the way the organization functions.

Component Name: [ViewStaffbyStaff](#)

-*Brief Description:

An ViewStaffbyStaff is a partial class which can be used to check and view their own Staff Details which were recorded at the very start through registration

Component Name: [ViewDetails](#)

Brief Description: An Admin gets the basic amenity to check User Details at his/her own will once He/She wants to see the record. It comes handy when one wants to check Details of the user.	
Methods()	Methods() Description
Button1_Click	View User Details along with Switch cases below.
viewCitizen()	To View Citizen's Record.
viewStaff()	To View Staff's Record.
viewCriminal	To View Criminal's Record.
viewVictim	To View victim's Record.
viewWitness();	To View Witness's Record.

Component Name: ViewUserDetailsbyStaff	
Brief Description: A Staff Official gets the basic amenity to check User Details at his/her own will once He/She wants to see the record. It comes handy when one wants to check Details of the user.	
Methods()	Methods() Description
Button1_Click	View User Details along with Switch cases below.
viewCitizen()	To View Citizen's Record.
viewStaff()	To View Staff's Record.
viewCriminal	To View Criminal's Record.
viewVictim	To View victim's Record.

Component Name: ViewFIR	
Brief Description: An Admin gets the basic amenity to view Registered FIR at his/her own will once He/She wants to see the record. It comes handy when one wants to check FIR Registered by the user.	
Methods	Methods() description

btnViewSelected_Click()	Button To Select FIR Id and display the Registered FIR.
btnViewAll_Click()	Button To Select FIR and display the Registered FIR.

Component Name: ViewFIRbyStaff	
Brief Description: Staff Official gets the basic amenity to view Registered FIR at his/her own will once He/She wants to see the record. It comes handy when one wants to check FIR Registered by the user.	
Methods	Methods() description
btnViewSelected_Click()	Button To Select FIR Id and display the Registered FIR.
btnViewAll_Click()	Button To Select FIR and display the Registered FIR.

Component Name: viewFIRstatusbyCitizen	
Brief Description: Citizen gets the basic amenity to view Registered FIR at his/her own will once He/She wants to see the record. It comes handy when one wants to check FIR Registered by the Citizen.	
Methods	Methods() description
btnView_Click()	<code>select fir_id,fir_status from FIR where cid</code>
btnView1_Click()	<code>select fir_id,fir_status from FIR where fir_id</code>

Component Name: ChangeCitizenPassword	
Brief Description: A Citizen has a Provision of Changing Password as it acts as step towards safer experience. To curb any phishing attack one should change his/her password once in every 3 months.	
Attributes	Attribute description

txtpass1	This will be the updated Password of the Citizen acting as the new Password. One must input this password in future attempts.
Txtpass	In order to see new password in effect, one must input the old Password. This also confirms that the user attempting to change password is a genuine user or not.
Methods()	Methods() Description
changePassword_Click()	Update Staff set pass.

Component Name: ChangeStaffPassword	
Brief Description: A Staff Official has a Provision of Changing Password as it acts as step towards safer experience throughout. To curb any phishing attack one should change his/her password once in every 3 months.	
-Attributes	Attribute description
txtpass1	This will be the updated Password of the Citizen acting as the new Password. One must input this password in future attempts.
Txtpass	In order to see new password in effect, one must input the old Password. This also confirms that the user attempting to change password is a genuine user or not.
Methods()	Methods() Description
changePassword_Click()	Update Staff set pass

Component Name: Signup	
Brief Description: A very basic Registration Process in which a Citizen/Staff Official/Admin has to add his/her own profile. A Citizen has to input his/her Username and password at each Login Attempt.	
Attributes	Attribute description
txtName	This will be displayed in the Profile Section. It acts as a personal identity.
txtAge	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtPhone	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtGender	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtAddress	This will be displayed in the Profile Section of each individual. It acts as a personal identity.
txtUserName	This will be the unique identity of the users. Username will be helping for the users to login to the system. This will be the primary identity of the authentication.
txtPassword	This will be secret identity of the user to logon to the system. Along with the username this secret ("Password") will be needed for a successful authentication.
Methods()	Methods() Description
SignUpBtn_Click()	Button to Signup and fill in the details
reset()	To nullify changes txtName, txtGender, txtAddress, txtUserName, txtPassword

Component Name: FillFeedback	
Brief Description: As we Know for an efficient functioning of any organization, we require Feedbacks from every user depicting how they feel regarding the System.	
Attributes	Attribute description
txtFeed	To fill the required feedback.
Methods()	Methods() Description
btnSubmit_Click()	Button to Delete an FIR.

Component Name: DeleteFIR	
-Brief Description: An Admin is given a provision to Delete Citizen's and Staff Official's Filed FIR be disposed at ones case gets Resolved.	
Attributes	Attribute description
dlistfird	To delete an FIR. To delete an FIR there and then.
Methods()	Methods() Description
btnDelete_Click()	Button to Delete an FIR.

Component Name: DeleteFIRbyStaff	
Brief Description: A Staff Official is given a provision to Delete Citizen's Filed FIR be disposed at ones the case gets Resolved.	

Attributes	Attribute description
dlistfirid	To delete an FIR there and then.
Methods()	Methods() Description
btnDelete_Click()	Button to Delete an FIR.

Component Name: DeleteUser	
Brief Description: An Admin gets the basic amenity to Delete any User Registered at his/her own will when as Required by the Police Headquarter. Normally, it happens when Crime Investigation gets over and Management wants to get rid of respective Records.	
Methods()	Methods() Description
btnDelete_Click()	Delete User along with Switch cases below
DeleteCitizen()	To Remove Citizen's Record.
DeleteStaff()	To Remove Staff's Record.
DeleteCriminal()	To Remove Criminal's Record.
DeleteVictim()	To Remove victim's Record.
DeleteWitness()	To Remove Witness's Record.

Component Name: RegisterFIRbyCitizen

Brief Description: At Citizens End, A Citizen also gets the basic amenity to check Registered FIR at his/her own will, once He/She gets an online FIR complaint. It comes handy when one wants to check FIR Record.	
Attributes	Attribute description
txtid	To display Registered FIR Id.
dlist_loc	To display Registered FIR location.
dlist_type	To display Registered FIR Type.
status	To display Registered FIR current Status.
txtdesp	To display Registered FIR Record.
Methods()	Methods() Description
Button1_Click1()	insert into FIR values
reset()	To nullify changes in Txtdesp. txtId, dlist_type, dlist_loc.

Component Name: RegisterFIRbyStaff	
Brief Description: At Staff End, A Staff Official also gets the basic amenity to check Registered FIR at his/her own will, once He/She gets an online FIR complaint. It comes handy when one wants to check FIR Record.	
Attributes	Attribute description
Txtid	To display Registered FIR Id.

dlist_loc	To display Registered FIR location.
dlist_type	To display Registered FIR Type.
status	To display Registered FIR current Status.
txtdesp	To display Registered FIR Record.
Methods()	Methods() Description
Button1_Click1()	insert into FIR values
reset()	To nullify changes in Txtdesp.

Component Name: UpdateStaffDetailsbyStaff	
Brief Description: At Staff's End, A Staff gets the basic amenity to UpdateStaff Details at his/her own will, once He/She wishes to. It comes handy when one wants to Update Crucial Own Details.	
Attributes	Attribute description
txtAge	To add Staff Official's and Citizen's Age.
txtAddress	To add Staff Official's and Citizen's Address.
txtPhone	To add Staff Official's and Citizen's Contact no.
Name	To add Staff Official's and Citizen's Name
Methods()	Methods() Description
Button1_Click1()	Update Staff & set age, address, phone, username.

reset()	To nullify changes in Address, Age, Phone.
---------	--

Component Name: UpdateCitizenDetails	
Brief Description: At Citizen's End, A Citizen gets the basic amenity to UpdateCitizen Details at his/her own will, once He/She wishes to. It comes handy when one wants to Update Crucial own Details.	
Attributes	Attribute description
txtAge	To add Citizen's Age.
txtAddress	To add Citizen's Address.
txtPhone	To add Citizen's Contact no.
txtUserName	To add Citizen's Name.
Methods()	Methods() Description
Button1_Click1()	Update Citizen and set age, address, phone, username.
reset()	To nullify changes in Address, Age, Phone.

Component Name: UpdateStaffDetailsbyStaff	
Brief Description: At Staff's End, A Staff gets the basic amenity to UpdateStaff Details at his/her own will, once He/She wishes to. It comes handy when one wants to Update Crucial Own Details.	

Attributes	Attribute description
txtAge	To add Staff Official's Age.
txtAddress	To add Staff Official's Address.
txtPhone	To add Staff Official's Contact no.
name	To add Staff Official's Name
Methods()	Methods() Description
Button1_Click1()	Update Staff set age, address, phone, username.
reset()	To nullify changes in Address, Age, Phone.

Component Name: UpdateFIR	
Brief Description: An Admin has a provision to Update Citizen's + Staff's Filed FIR.	
Attributes	Attribute description
dlist_status	To update an FIR there and then if any.
Methods()	Methods() Description
btnUpdate_Click()	Update FIR along with setting fir_status.

Component Name: UpdateFIRbyStaff
--

Brief Description: Staff official has a provision to Update Citizen's Filed FIR.	
Attributes	Attribute description
dlist_status	To update an FIR there and then if any.
Methods()	Methods() Description
btnUpdate_Click()	It allows to click Update FIR by Staff.

Component Name: UpdateUserDetails	
Brief Description: Every User is given a facility to update Citizen's and Staff's Details as per their requirement at his/her own will. It comes handy when one wants to Update Own Details.	
Attributes	Attribute description
txtAge	To add Staff Official's Age.
txtAddress	To add Staff Official's Address.
txtPhone	To add Staff Official's Contact no.
name	To add Staff Official's Name
Methods()	Methods() Description
Button1_Click1()	It allows to click Update Citizen or Update Staff.
updateCitizen()	It allows Admin to updateCitizen's Detail
updateStaff()	It allows Admin to updateStaff's Detail

5.0 Design decisions and tradeoffs

Our team was successful in implementing the project to a basic working model. We've been able in meeting the various basic requirements needed by an ONLINE CRIME SCENE INVESTIGATION SYSTEM , that were listed by our team and our clients. There were various limiting factors that came up during its implementation. They are listed below:

1) Time:

- it is a world wide phenomena that the most occurring limiting constraint is time that hampers the projects perfection. We as students had to manage it with the other courses that are part of our semester, hence giving equal time to all subjects became a constraint.
- Our team felt if time would have been extended to make the system we could have worked a bit more on our basic level model.

2) Interactions

Interaction with the staff members became a constraint as we couldnt meet them during their busy schedule. Even we had to attend all our classes as we in no manner couldve missed them. Hence finding a suitable free time for interaction with the staff where both our timetables would match rather than clash was a limiting factor interms of speed of development of the software/web application.

3)Client side issue

Inorder to access the server set up by the developers , the client side would require a stable fast internet connection and a suitable system with an updated web browser to access the web app. They should also have updated visual c++ redistributable to access/ run the web app smoothly.

4) Team members technical skills (internal problem)

During the start of the project the team members discussed about the requirements to build such a robust system and many of the members werent having the desired technical skills which were needed out of them. This led to increased delay in the making of project.

6.0 Pseudocode for components

Pseudo code

Class Addcitizendetails()

protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void SignUpBtn_Click	{ cmd = " sqlcommand("insert into ..."); Sqlcom.open(); i = cmd.execute(); if(i>0) //records saved Sqlcon.close(); Reset();
public void reset	// Resetting all the values txtName.Text = ""; txtAge.Text = ""; txtPhone.Text = ""; txtGender.Text = ""; txtAddress.Text = ""; txtPassword.Text = ""; txtUserName.Text = ""; txtPassword.Text = "";

Class Addstaffdetails()	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void SignUpBtn_Click	{ cmd = " sqlcommand("insert into ..."); Sqlcom.open(); i = cmd.execute(); if(i>0) //records saved Sqlcon.close(); Reset();
public void reset	// Resetting all the values txtName.Text = ""; txtAge.Text = ""; txtPhone.Text = ""; txtGender.Text = ""; txtAddress.Text = ""; txtPassword.Text = ""; txtUserName.Text = ""; txtPassword.Text = "";

class Admin_Login	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void Button1_Click	{ if (txtname.txt = "" && txtpassword.txt = "") { alert("fill the form);} Else{ String = "select * from where.... "; //check in database Sqlcon.open(); //execute and store in rd If(rd.read()) Redirect("adminprofile.aspx"); Else Alert("invalid credentials");

class ChangeCitizenPassword	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void changePassword_Click	{ sqlcon.open(); Name = session["un"].ToString(); If(txtpass.txt== txtpass1.txt) { Cmd = new sqlcommand("update citizen set pass"); Cmd.executeNonQuery(); Lb1.text = "password changed"; Sqlcon.close;

class ChangeStaffPassword

protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void changePassword_Click	{ sqlcon.open(); Name = session["un"].ToString(); If(txtpass.txt== txtpass1.txt) { Cmd = new sqlcommand("update citizen set pass"); Cmd.executeNonQuery(); Lb1.text = "password changed"; Sqlcon.close;

class Citizen_Login	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void Button1_Click	{ if (txtname.txt = "" && txtpassword.txt = "") { alert("fill the form);} Else{ String = "select * from where.... "; //check in database Sqlcon.open(); //execute and store in rd If(rd.read()) Redirect("adminprofile.aspx"); Else Alert("invalid credentials");

class DeleteFIR	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }

protected void btnDelete_Click	{sqlcon.open(); Cmd = new sqlcommand("delete from where...."); cmd.ExecuteNonQuery(); SqlCon.Close(); lbl.Text = "Deletion Successfull";
class DeleteFIRbyStaff	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void btnDelete_Click	{sqlcon.open(); Cmd = new sqlcommand("delete from where...."); cmd.ExecuteNonQuery(); SqlCon.Close(); lbl.Text = "Deletion Successfull";

class DeleteUser	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void btndelete_Click	//switching to required method switch (n) { case 1: deleteCitizen(); break; case 2: deleteStaff(); break; case 3: deleteCriminal(); break; case 4: deleteVictim(); break; case 5: deleteWitness(); break; default: break;

	<pre> }</pre>
<pre> public void deleteCitizen()</pre>	<pre> Sqlcon.open(); Cmd = new sqlcomand("delete from where...."); cmd.ExecuteNonQuery(); SqlCon.Close(); txtUserName.Text = ""; lbl.Text = "Deletion Successfull";</pre>
<pre> public void deleteCriminal()</pre>	<pre> Sqlcon.open(); Cmd = new sqlcomand("delete from where...."); cmd.ExecuteNonQuery(); SqlCon.Close(); txtUserName.Text = ""; lbl.Text = "Deletion Successfull";</pre>
<pre> public void deleteVictim()</pre>	<pre> Sqlcon.open(); Cmd = new sqlcomand("delete from where...."); cmd.ExecuteNonQuery(); SqlCon.Close(); txtUserName.Text = ""; lbl.Text = "Deletion Successfull";</pre>
<pre> public void deleteWitness()</pre>	<pre> Sqlcon.open(); Cmd = new sqlcomand("delete from where...."); cmd.ExecuteNonQuery(); SqlCon.Close(); txtUserName.Text = ""; lbl.Text = "Deletion Successfull";</pre>

```

class FillFeedback
```

protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void btnSubmit_Click	{cmd = new sqlcommand("insert into"); SqlCon.Open(); int i = cmd.ExecuteNonQuery(); if(i>0) //thanks for feedback;

class Logout	
protected void Page_Load	Session.Abandon(); Response.Redirect("Home.aspx");

class RegisterFIR	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }

protected void btnregister_Click	{ sqlcon.open(); cmd = " sqlcommand("insert into ..."); i = cmd.execute(); ilbl.Text = "FIR registered successfully"; //records saved Sqlcon.close(); Reset();
public void reset()	txtid.Text = ""; txt desp.Text = ""; dlist_type.SelectedItem.Value = ""; dlist_loc.SelectedItem.Value = "";

class RegisterFIRbyCitizen	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void btnregister_Click	{ sqlcon.open(); cmd = " sqlcommand("insert into ..."); i = cmd.execute(); ilbl.Text = "FIR registered successfully"; //records saved Sqlcon.close(); Reset();
public void reset()	txtid.Text = ""; txt desp.Text = ""; dlist_type.SelectedItem.Value = ""; dlist_loc.SelectedItem.Value = "";

class RegisterFIRbyStaff

protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void btnregister_Click	{ sqlcon.open(); cmd = " sqlcommand("insert into ..."); i = cmd.execute(); ilbl.Text = "FIR registered successfully"; //records saved Sqlcon.close(); Reset();
public void reset()	txtid.Text = ""; txt desp.Text = ""; dlist_type.SelectedItem.Value = ""; dlist_loc.SelectedItem.Value = "";

class Signup	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void SignUpBtn_Click	{ cmd = " sqlcommand("insert into ..."); Sqlcom.open(); i = cmd.execute(); if(i>0) //records saved Sqlcon.close(); Reset();
public void reset	// Resetting all the values txtName.Text = ""; txtAge.Text = ""; txtPhone.Text = ""; txtGender.Text = ""; txtAddress.Text = ""; txtPassword.Text = ""; txtUserName.Text = ""; txtPassword.Text = "";

class Staff_Login

protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void Button1_Click	{ if (txtname.txt = "" && txtpassword.txt = "") { alert("fill the form");} Else{ String = "select * from where.... "; //check in database Sqlcon.open(); //execute and store in rd If(rd.read()) Redirect("adminprofile.aspx"); Else Alert("invalid credentials");
class UpdateCitizenDetails	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void Button1_Click1	{ sqlcon.open(); Cmd = new sqlcommand("update citizen set pass"); Cmd.executeNonQuery(); Lb1.text = "updated"; Sqlcon.close; Reset();
public void reset()	txtAddress.Text = ""; txtAge.Text = ""; txtPhone.Text = ""; txtUserName.Text = "";

class UpdateFIR	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }

void btnUpdate_Click	{ sqlcon.open(); Cmd = new sqlcommand("update citizen set pass"); Cmd.executeNonQuery(); Lb1.text = "FIR status updated sucessfully"; Sqlcon.close;
----------------------	---

class UpdateFIRbyStaff	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
void btnUpdate_Click	{ sqlcon.open(); Cmd = new sqlcommand("update citizen set pass"); Cmd.executeNonQuery(); Lb1.text = "FIR status updated sucessfully"; Sqlcon.close;

class UpdateStaffDetailsbyStaff	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void Button1_Click1	{ sqlcon.open(); Name = session["un"].ToString(); Cmd = new sqlcommand("update staff set pass"); Cmd.executeNonQuery(); SqlCon.Close(); Lb1.text = "Updated"; Sqlcon.close;
public void reset()	txtAddress.Text = ""; txtAge.Text = ""; txtPhone.Text = "";

Class UpdateUserDetails

protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void Button1_Click1	//switching to required method switch (n) { case 1: updateCitizen(); break; case 2: updateStaff(); break;}
public void updateCitizen()	Sqlcon.open(); Cmd = new sqlcomand("update from where...."); cmd.ExecuteNonQuery(); SqlCon.Close();
public void updateStaff()	Sqlcon.open(); Cmd = new sqlcomand("update from where...."); cmd.ExecuteNonQuery(); SqlCon.Close();

class ViewDetails	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void Button1_Click	//switching to required method switch (n) { case 1: viewCitizen(); break; case 2: viewStaff(); break;

	<pre> case 3: viewCriminal(); break; case 4: viewVictim(); break; case 5: viewWitness(); break; default: break;} </pre>
public void viewCitizen()	<pre> Sqlcon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "Citizen"); GridView1.DataSource = ds.Tables["Citizen"]; GridView1.DataBind(); SqlCon.Close(); </pre>
public void viewStaff()	<pre> Sqlcon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "staff"); GridView1.DataSource = ds.Tables["staff"]; GridView1.DataBind(); SqlCon.Close(); </pre>
public void viewCriminal()	<pre> Sqlcon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "criminal"); GridView1.DataSource = ds.Tables["criminal"]; GridView1.DataBind(); SqlCon.Close(); </pre>
public void viewVictim()	<pre> Sqlcon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "victim"); GridView1.DataSource = ds.Tables["victim"]; GridView1.DataBind(); SqlCon.Close(); </pre>

public void viewWitness()	<pre> Sqlcon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "witness"); GridView1.DataSource = ds.Tables["witness"]; GridView1.DataBind(); SqlCon.Close(); </pre>
---------------------------	---

class ViewFIR	
protected void Page_Load	<pre> { sqlconnection = "....." //connecting variables Sql con = new sql connection; } </pre>
protected void btnViewSelected_Click	<pre> //get list as filtered { cmd = new sqlcommand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "FIR"); //combine them to grids </pre>
protected void btnViewAll_Click	<pre> //get all list from db { cmd = new sqlcommand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "FIR"); //combine them to grids </pre>

class ViewFIRbyStaff	
protected void Page_Load	<pre> { sqlconnection = "....." //connecting variables Sql con = new sql connection; } </pre>

protected void btnViewSelected_Click	//get list as filtered { cmd = new sqlcommand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "FIR"); //combine them to grids
protected void btnViewAll_Click	//get all list from db { cmd = new sqlcommand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "FIR"); //combine them to grids

class viewFIRstatusbyCitizen	
protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void btnViewSelected_Click	//get list as filtered { cmd = new sqlcommand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "FIR"); //combine them to grids
protected void btnViewAll_Click	//get all list from db { cmd = new sqlcommand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "FIR"); //combine them to grids

class ViewUserDetailsbyStaff

protected void Page_Load	{ sqlconnection = "....." //connecting variables Sql con = new sql connection; }
protected void Button1_Click	//switching to required method switch (n) { case 1: viewCitizen(); break; case 2: viewStaff(); break; case 3: viewCriminal(); break; case 4: viewVictim(); break; case 5: viewWitness(); break; default: break;}
public void viewCitizen()	Sqlcon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "Citizen"); GridView1.DataSource = ds.Tables["Citizen"]; GridView1.DataBind(); SqlCon.Close();
public void viewStaff()	Sqlcon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "staff"); GridView1.DataSource = ds.Tables["staff"]; GridView1.DataBind(); SqlCon.Close();

public void viewCriminal()	SqlCon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "criminal"); GridView1.DataSource = ds.Tables["criminal"]; GridView1.DataBind(); SqlCon.Close();
public void viewVictim()	SqlCon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "victim"); GridView1.DataSource = ds.Tables["victim"]; GridView1.DataBind(); SqlCon.Close();
public void viewWitness()	SqlCon.open(); Cmd = new sqlcomand("select * from where...."); SqlDataAdapter adp = new SqlDataAdapter(cmd); DataSet ds = new DataSet(); adp.Fill(ds, "witness"); GridView1.DataSource = ds.Tables["witness"]; GridView1.DataBind(); SqlCon.Close();

7.0 Appendices (if any)

N.A