

CONTENTS

<i>Contents</i>	1
<i>Introduction</i> :	7
<i>Objective</i> :.....	8
<i>System Analysis</i>	8
<i>Identification of Need</i> :.....	8
<i>Preliminary Investigation</i> :.....	8
<i>Feasibility Study</i> :	9
<i>Project Planning & Scheduling</i> :	9
<i>Gantt chart</i>	9
<i>Tracking Gantt</i>	9
<i>Pert Chart</i>	10
<i>Software requirement specifications (SRS)</i> :	10
<i>Functional Requirement</i>	11
<i>Non-functional Requirements</i>	13
<i>Software Engineering Paradigm applied</i>	13
<i>Data models</i>	15
<i>Context Diagram</i>	15
<i>0-Level DFD</i>	15
<i>1-Level DFD</i>	16
<i>2-Level DFD</i>	17
<i>Sequence diagrams</i>	17
<i>Register images</i>	17
<i>images PROCESSING</i>	18
<i>Entity Relationship Model</i>	19
<i>Class Diagrams</i>	20
<i>Activity Diagrams</i>	21
<i>System Design</i>	24

<i>Modularisation details</i>	24
<i>Database & Table Details</i>	24
<i>COMPLETE DATA STRUCTURE</i>	25
<i>Data integrity and constraints</i>	25
<i>Entity integrity</i>	25
<i>Referential Integrity</i>	26
<i>Domain Integrity</i>	26
<i>User Defined Integrity</i>	26
<i>Database design</i>	26
<i>Table: user</i>	26
<i>Table: admin</i>	26
<i>Table: image</i>	27
<i>Table: result</i>	27
<i>User Interface Design</i>	27
<i>Main window</i>	27
<i>Recognize face</i>	28
<i>Register Image</i>	28
<i>CREATE ACCOUNT</i>	29
<i>GOOGLE OPEN ID</i>	30
<i>Test Cases (Unit Test Cases and System Test Cases)</i>	30
<i>Unit Test Cases</i>	30
<i>System Test Cases</i>	33
<i>Coding</i>	35
<i>Complete Project Coding</i>	35
<i>FRS GUI Design Coding: FRSGUI</i>	35
<i>GUI Style : FRSSyles</i>	41
<i>Business Logic – Matlab</i>	48
<i>Database connector: FRSDb</i>	66

<i>Database Classes : FRSDData.....</i>	<i>73</i>
<i>Comments and Description of Coding segments.....</i>	<i>74</i>
<i>Code Commenting.....</i>	<i>74</i>
<i>Description of coding.....</i>	<i>76</i>
<i>Standardization of the coding.....</i>	<i>77</i>
<i>Code Efficiency</i>	<i>78</i>
<i>Error handling</i>	<i>79</i>
<i>Exceptions Overview</i>	<i>79</i>
<i>Validation checks</i>	<i>80</i>
<i>Allowed character checks</i>	<i>80</i>
<i>Batch totals</i>	<i>80</i>
<i>Cardinality check.....</i>	<i>80</i>
<i>Check digits.....</i>	<i>81</i>
<i>Consistency checks.....</i>	<i>81</i>
<i>Control totals.....</i>	<i>81</i>
<i>Cross-system consistency checks</i>	<i>81</i>
<i>Data type checks.....</i>	<i>81</i>
<i>File existence check</i>	<i>81</i>
<i>Format or picture check.....</i>	<i>81</i>
<i>Hash totals.....</i>	<i>82</i>
<i>Limit check.....</i>	<i>82</i>
<i>Logic check</i>	<i>82</i>
<i>Presence check.....</i>	<i>82</i>
<i>Range check.....</i>	<i>82</i>
<i>Referential integrity</i>	<i>82</i>
<i>Spelling and grammar check</i>	<i>82</i>
<i>Uniqueness check.....</i>	<i>82</i>
<i>Table Look Up Check</i>	<i>82</i>

<i>Testing</i>	<i>83</i>
<i>Testing techniques and testing strategies used</i>	<i>83</i>
<i>Database & Data Integrity Testing.....</i>	<i>83</i>
<i>Functional Testing:.....</i>	<i>83</i>
<i>Regression Testing:</i>	<i>84</i>
<i>User Interface Testing:</i>	<i>84</i>
<i>Performance Profiling:</i>	<i>85</i>
<i>Load Testing:</i>	<i>86</i>
<i>Stress Testing:.....</i>	<i>86</i>
<i>Volume Testing:</i>	<i>87</i>
<i>Security & Access Control Testing:</i>	<i>87</i>
<i>Failover & Recovery Testing:.....</i>	<i>87</i>
<i>Configuration Testing:.....</i>	<i>87</i>
<i>Installation/Deploy & Back out Testing:.....</i>	<i>88</i>
<i>Post Production Testing:.....</i>	<i>88</i>
<i>Unit Testing:</i>	<i>88</i>
<i>Smoke Testing:.....</i>	<i>88</i>
<i>Data Migration Testing:</i>	<i>89</i>
<i>Testing Plan used</i>	<i>90</i>
<i>Cause of Testing:.....</i>	<i>90</i>
<i>Unit testing includes the following:.....</i>	<i>91</i>
<i>System testing includes the following:</i>	<i>91</i>
<i>The following points should be remembered primarily:</i>	<i>92</i>
<i>Any engineered product (and most other things) can be tested in one of two ways:</i>	<i>92</i>
<i>White-box testing:.....</i>	<i>93</i>
<i>Basis Path Testing:</i>	<i>94</i>
<i>Black-box Testing:.....</i>	<i>94</i>
<i>Test reports for Unit Test Cases and System Test Cases</i>	<i>96</i>

<i>Test reports for Unit Test Cases</i>	96
<i>Test reports for System Test Cases</i>	98
<i>Debugging and Code improvement:</i>	99
<i>Create a Sample with the Debug Class</i>	101
<i>Using the Trace Class</i>	102
<i>Verify That It Works</i>	103
<i>Complete Code Listing</i>	106
<i>Troubleshoot</i>	108
<i>System Security measures:</i>	108
<i>Database/data security:</i>	108
<i>Creation of User profiles and access rights</i>	108
<i>Cost Estimation of the Project along with Cost Estimation Model</i>	109
<i>Estimation of development effort</i>	111
<i>Estimation of development time</i>	111
<i>Reports</i>	111
<i>Future scope and further enhancement of the Project</i>	112
<i>Bibliography</i>	112
<i>Website</i>	112
<i>Books</i>	114
<i>Appendices</i>	114
<i>IDE Used:</i>	114
<i>Visual Studio 2010</i>	114
<i>MATLAB</i>	116
<i>Front End - WPF (Windows Presentation Framework)</i>	116
<i>Programming with wpf</i>	117
<i>Markup & code-behind</i>	118
<i>security</i>	118
<i>controls</i>	118

<i>Wpf controls by function</i>	<i>118</i>
<i>layout</i>	<i>119</i>
<i>Graphics</i>	<i>120</i>
<i>Extensible application Markup Language (XAML).....</i>	<i>120</i>
<i>Programming Framework.....</i>	<i>121</i>
<i>.NET 4.5</i>	<i>121</i>
<i>Database/backend:</i>	<i>130</i>
<i>MySQL</i>	<i>130</i>
<i>ide for Database.....</i>	<i>133</i>
<i>MySQL workbench</i>	<i>133</i>
<i>Programming Language.....</i>	<i>135</i>
<i>C# - C sharp.....</i>	<i>135</i>
<i>MATLAB.....</i>	<i>138</i>
<i>Dia for Diagram Drawing &Modeling.....</i>	<i>138</i>
<i>Google Spreadsheet Interface:.....</i>	<i>139</i>
<i>Cacoo:: online drawing tool.....</i>	<i>140</i>
<i>Creating Diagrams</i>	<i>140</i>
<i>Collaboration</i>	<i>140</i>
<i>Sharing Diagrams.....</i>	<i>140</i>
<i>Managing Diagrams</i>	<i>141</i>
<i>Languages and Time Zones</i>	<i>141</i>
<i>Security.....</i>	<i>141</i>
<i>API.....</i>	<i>141</i>
<i>Version Control System : GitHub</i>	<i>142</i>
<i>Description.....</i>	<i>142</i>
<i>Limitations and constraints</i>	<i>142</i>
<i>Glossary.</i>	<i>142</i>

INTRODUCTION:

Face modeling for recognition of people is a challenging problem which has received much attention during recent years due to its many applications in different fields. Face recognition has become a very active area of research in recent years mainly due to increasing security demands and its potential commercial and law enforcement applications. The last decade has shown dramatic progress in this area, with emphasis on such applications as human-computer interaction (HCI), biometric analysis, content-based coding of images and videos, and surveillance.

Face recognition and detection is a reliable technique to authenticate a user's identification. So facial recognition system and detection is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source any source. One of the ways to do this is by comparing selected facial features from the image and a facial database.

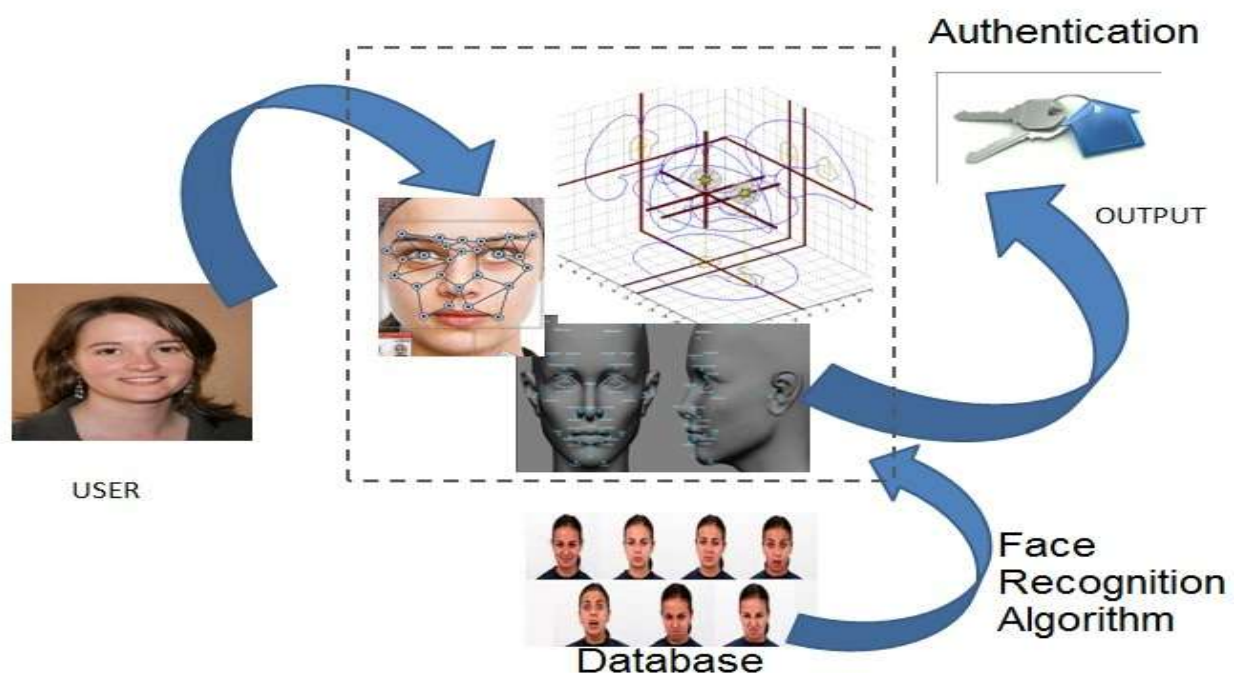


Fig: Overview of Face Recognition System

Face Recognition System will enable users to authenticate themselves using their facial images which will be compared to existing database using Gabor Analysis Algorithm.



Face Recognition Algorithm will follow below mentioned steps to identify the face.

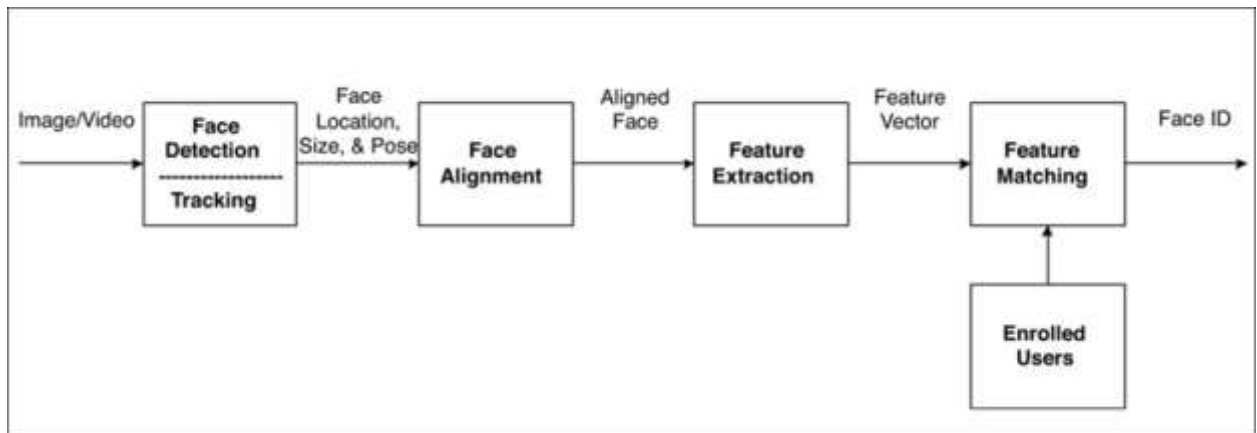


Fig: Flow of Face Recognition Process

OBJECTIVE:

It's a true challenge to build an automated system which equals human ability to recognize faces. Although humans are quite good identifying known faces, we are not very skilled when we must deal with a large amount of unknown faces. The computers, with an almost limitless memory and computational speed, should overcome human limitations. Face Recognition System will facilitate face recognition and enable user authentication using this.

Face recognition and detection typically used in security system and can be compared to other biometrics such as face or eye iris recognition systems.

Admin has to register their face images from different angle and images will be stored in database. Whenever a user wants to authenticate, the Face Recognition System will take picture of the user and compare with existing database using Gabor Analysis Algorithm.

There are following feature in face recognition system:

- Recognition from outdoor facial images.
- Recognition from non-frontal facial images.
- Preprocesses the captured images (removes background, scales size, and so on)
- Add , modify users
- Authenticate user based on registered user facial features
- Integration with Google Open ID
- Admin user account & normal user account

SYSTEM ANALYSIS

IDENTIFICATION OF NEED:

It is hard to memorise password or other security information. When we want to login to any app or web site through id paddwod it is boaring. But if we secure with all id password with our pace or physical identification we can protect all the unauthorised usesges of our personal data. On the other hand we can track others face for security reason. From this requirement we started to develop this techonological equipment.

PRELIMINARY INVESTIGATION:

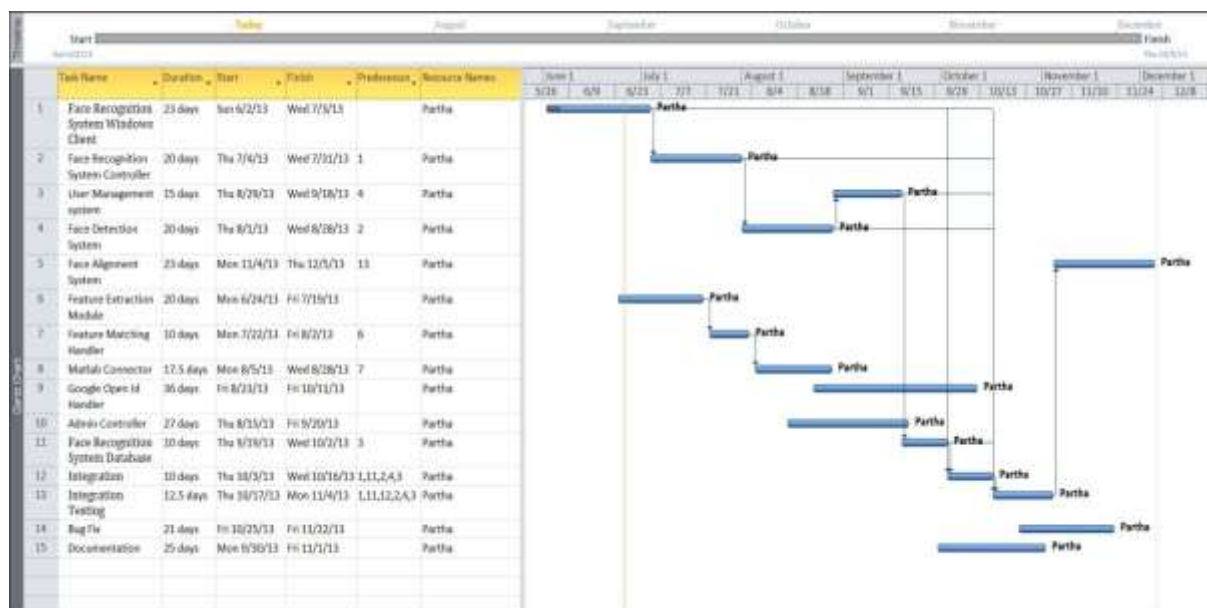
I spoke with many of my friends who helped me to thought about these technology and the features. I thought a desktop application could be developed to minimize the memorisation effort. I then started gathering opinion from IT professionals and friends. I gathered all the important points including my own opinion and decided to develop Face Recognizer.

FEASIBILITY STUDY:

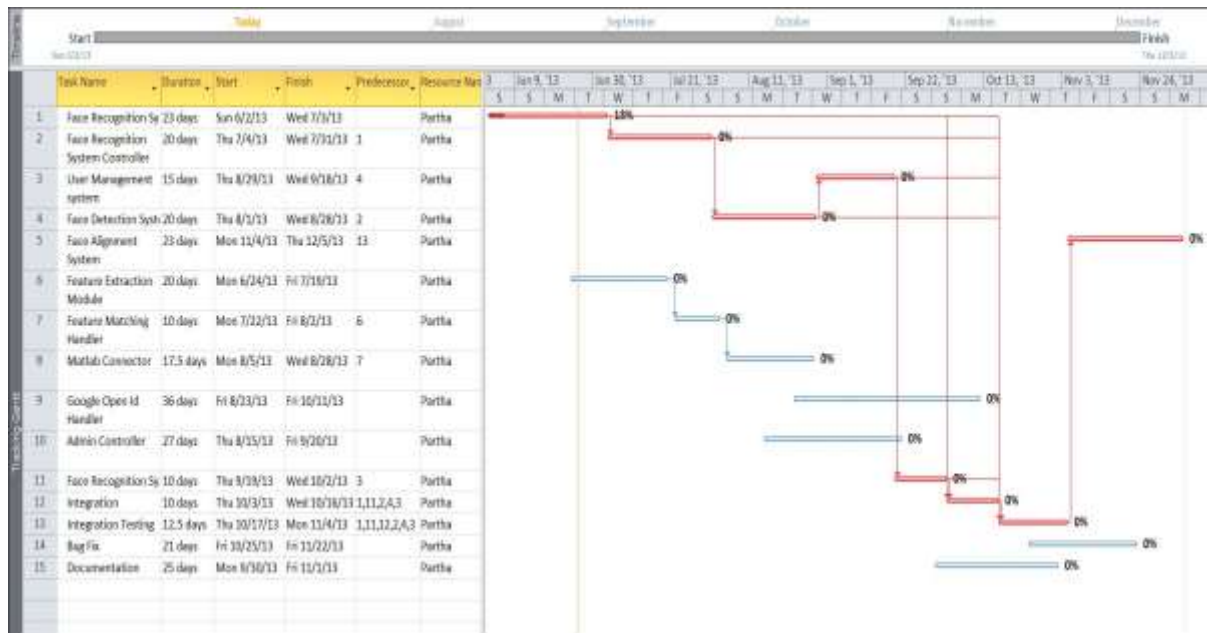
Facing problem about forgetting password or user id is a common problem among us. People would love an application that would memorise their physical identity for authentication. I have decided to provide a face tracking system to identify the registered user. So, undoubtedly it is going to be a popular desktop application.

PROJECT PLANNING & SCHEDULING:

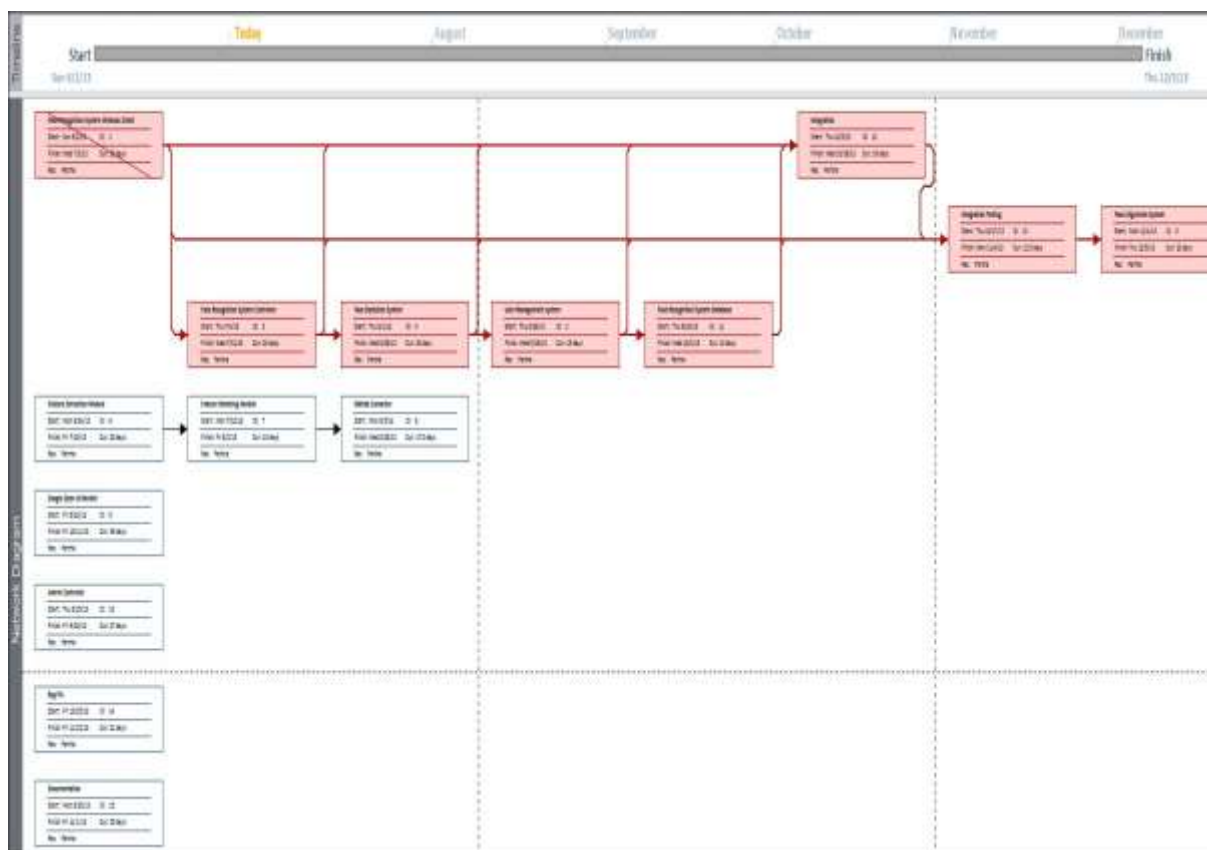
GANTT CHART



TRACKING GANTT



PERT CHART



SOFTWARE REQUIREMENT SPECIFICATIONS (SRS):

FUNCTIONAL REQUIREMENT

REGISTER USER IMAGES

INTRODUCTION

Admin has to register their face images from different angles and images will be stored in database.

INPUT

Admin will be storing ten photos for each person from different predefined angles.

PROCESSING

These images will be stored in database for processing.

OUTPUT

Images stored successfully to the database and ready to be accessed by face recognition system.

CREATE USER ACCOUNT FOR AUTHENTICATION

INTRODUCTION:

Create account for a new User.

INPUT:

Relevant User data like user name, images, etc.

PROCESSING:

The Face Recognition will create a new user entry.

OUTPUT:

The face recognition will generate a user to reach his face identification.

FACE RECOGNITION

INTRODUCTION:

Face recognition users has to provide an image which will be matched with existing image stored in database using face recognition techniques.

INPUT:

User images captured from different resources like video, images.

PROCESS:

Input image which will be matched with existing image stored in database using face recognition techniques.

OUTPUT:

User images will be matched and authenticated by admin.

FACE RECOGNITION FROM IMAGES OF VARYING ANGLES

INTRODUCTION:

Face recognition users has to provide images from any of the predefined angles which will be matched with existing image stored in database using face recognition techniques.

INPUT:

User images captured from webcam

PROCESS:

Input image which will be matched with existing image stored in database using face recognition techniques.

OUTPUT:

User images will be matched and authenticated.

INTEGRATION WITH GOOGLE OPEN ID

INTRODUCTION:

Face recognition users has to provide images from any of the predefined angles which will be matched with existing image stored in database using face recognition techniques and authenticate users corresponding Google open id account.

INPUT:

User images captured from webcam

PROCESS:

Input image which will be matched with existing image stored in database using face recognition techniques and retrieve the corresponding Google open id from the database.

OUTPUT:

User images will be matched and authenticated. Thus the user will be login to Google account without entering password manually.

NON-FUNCTIONAL REQUIREMENTS

This project is intended to meet the following nonfunctional requirements: -

- This face recognition software should be available on the Internet, to enable the users to use , download it any time.
- The program should be platform independent.
- Accuracy should be more
- Authentication process should not take more than one min.

SOFTWARE ENGINEERING PARADIGM APPLIED

We have followed agile version of Model Driven Development (MDD). As the name implies, AMDD is the agile version of Model Driven Development (MDD). MDD is an approach to software development where extensive models are created before source code is written. A primary example of MDD is the Object Management Group (OMG)'s Model Driven Architecture (MDA) standard. With MDD a serial approach to development is often taken, MDD is quite popular with traditionalists, although as the RUP/EUP shows it is possible to take an iterative approach with MDD. The difference with AMDD is that instead of creating extensive models before writing source code you instead create agile models which are just barely good enough that drive your overall development efforts. AMDD is a critical strategy for scaling agile software development beyond the small, co-located team approach that we saw during the first stage of agile adoption.

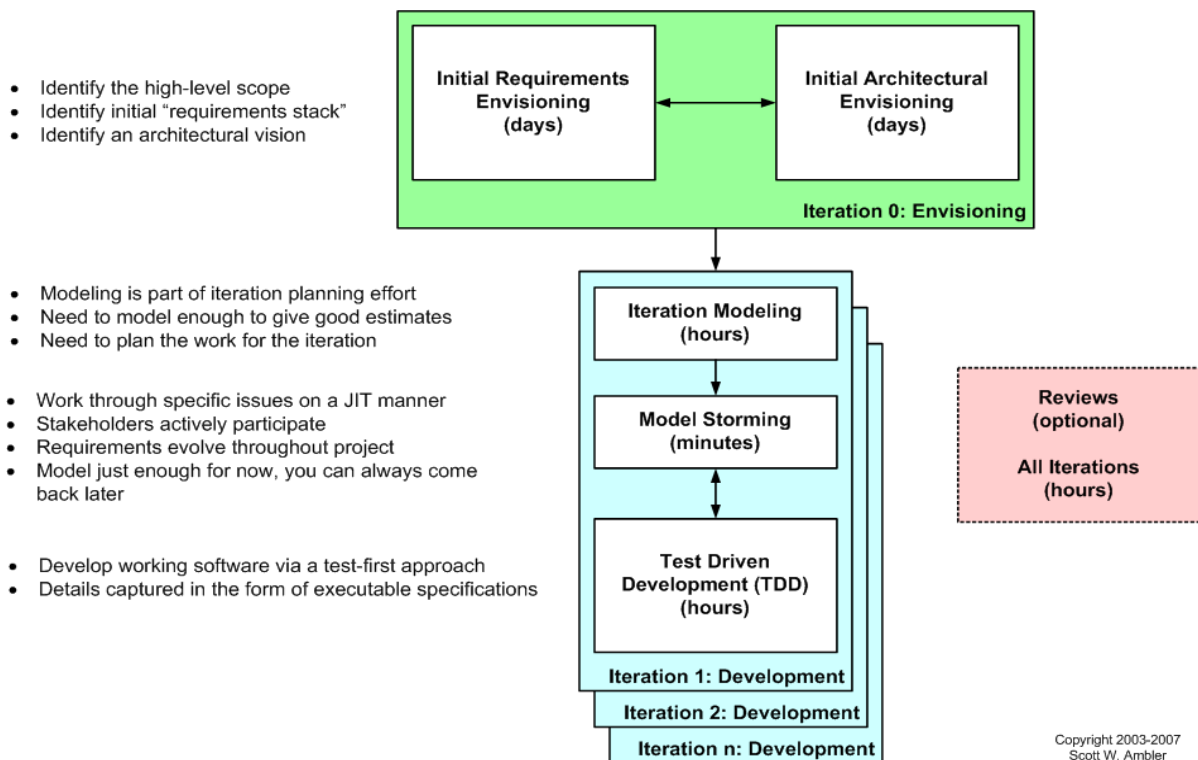


Figure 1: The AMDD lifecycle: Modeling activities throughout the lifecycle of a project

Above Figure depicts a high-level lifecycle for AMDD for the release of a system. First, let's start with how to read the diagram. Each box represents a development activity. The envisioning includes two main sub-activities, initial requirements envisioning and initial architecture envisioning. These are done during iteration 0, iteration being another term for cycle or sprint. "Iteration 0" is a common term for the first iteration before you start into development iterations, which are iterations one and beyond (for that release). The other activities – iteration modeling, model storming, reviews, and implementation – potentially occur during any iteration, including iteration 0. The time indicated in each box represents the length of an average session: perhaps you'll model for a few minutes then code for several hours. I'll discuss timing issues in more detail below..

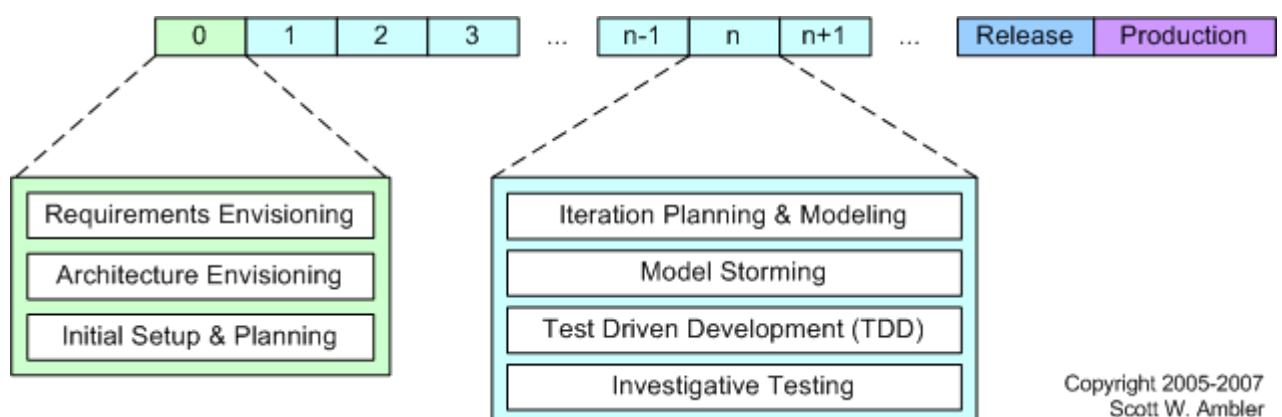
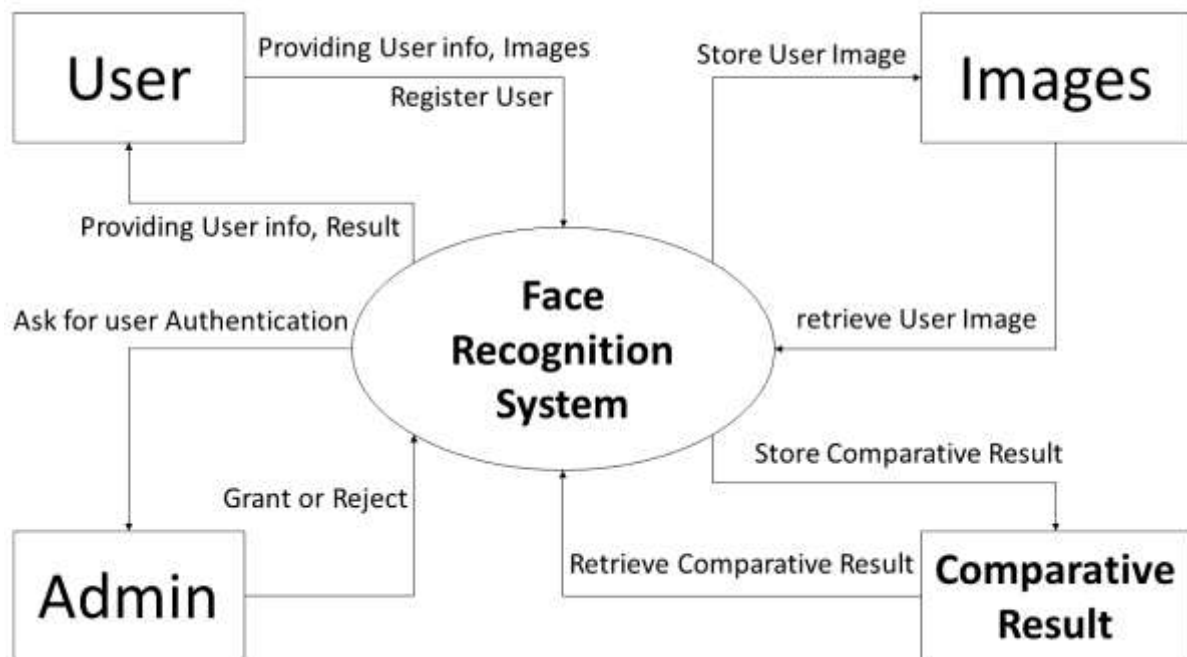


Figure 2AMDD Through the Agile Development Lifecycle.

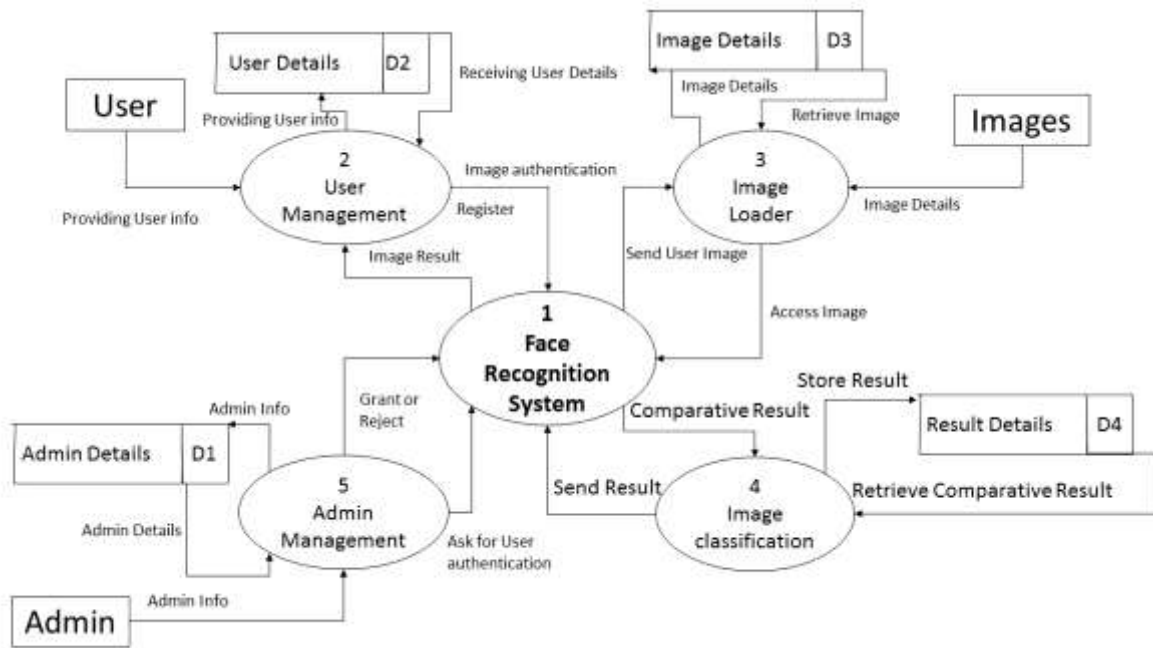
Above Figure depicts how the AMDD activities fit into the various iterations of the agile software development lifecycle. It's simply another way to show that an agile project begins with some initial modelling and that modelling still occurs in each construction's iteration.

DATA MODELS

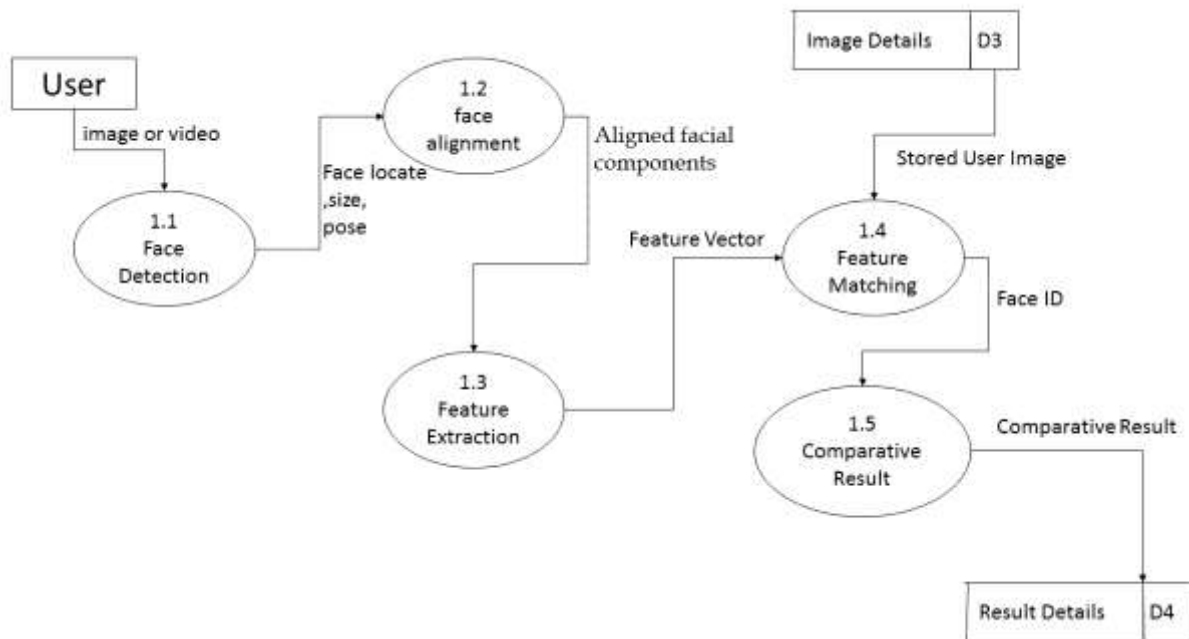
CONTEXT DIAGRAM

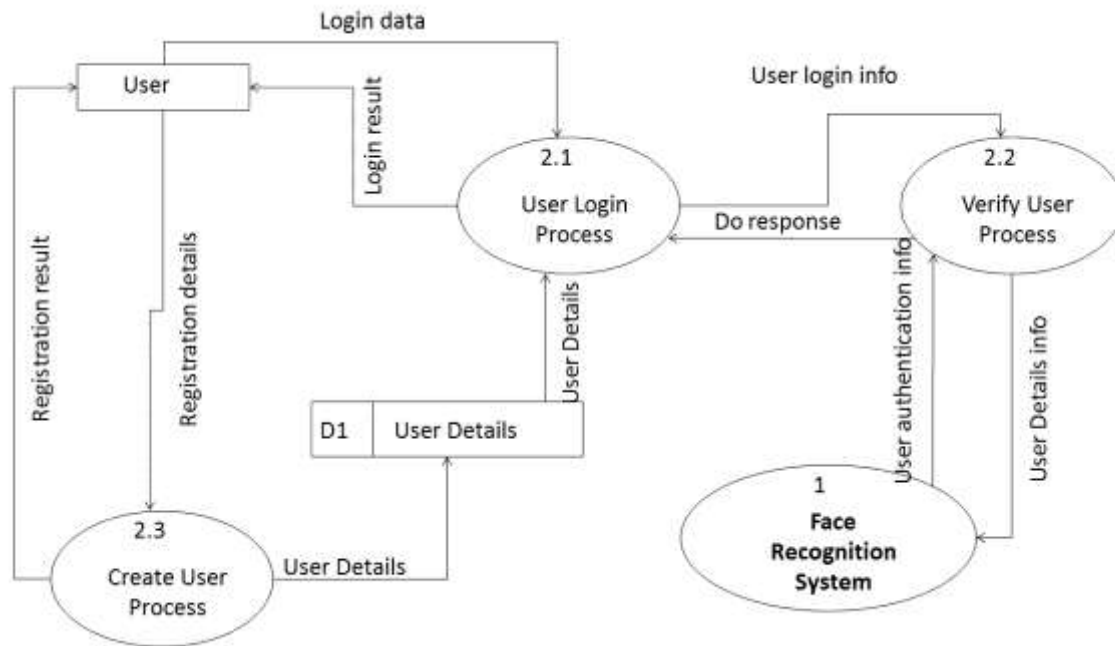


0-LEVEL DFD

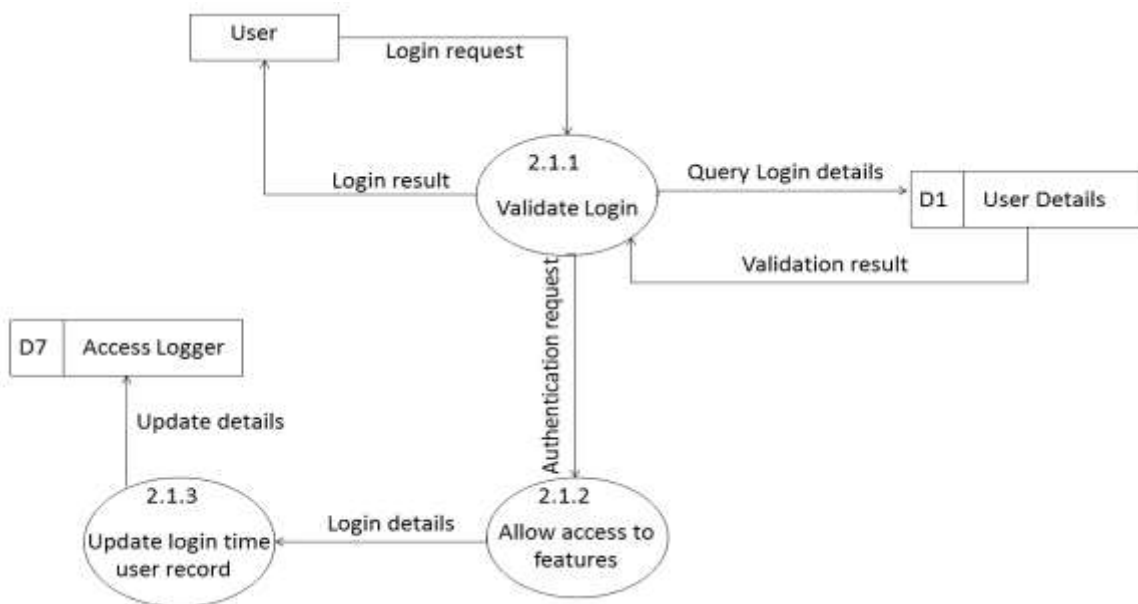


1-LEVEL DFD



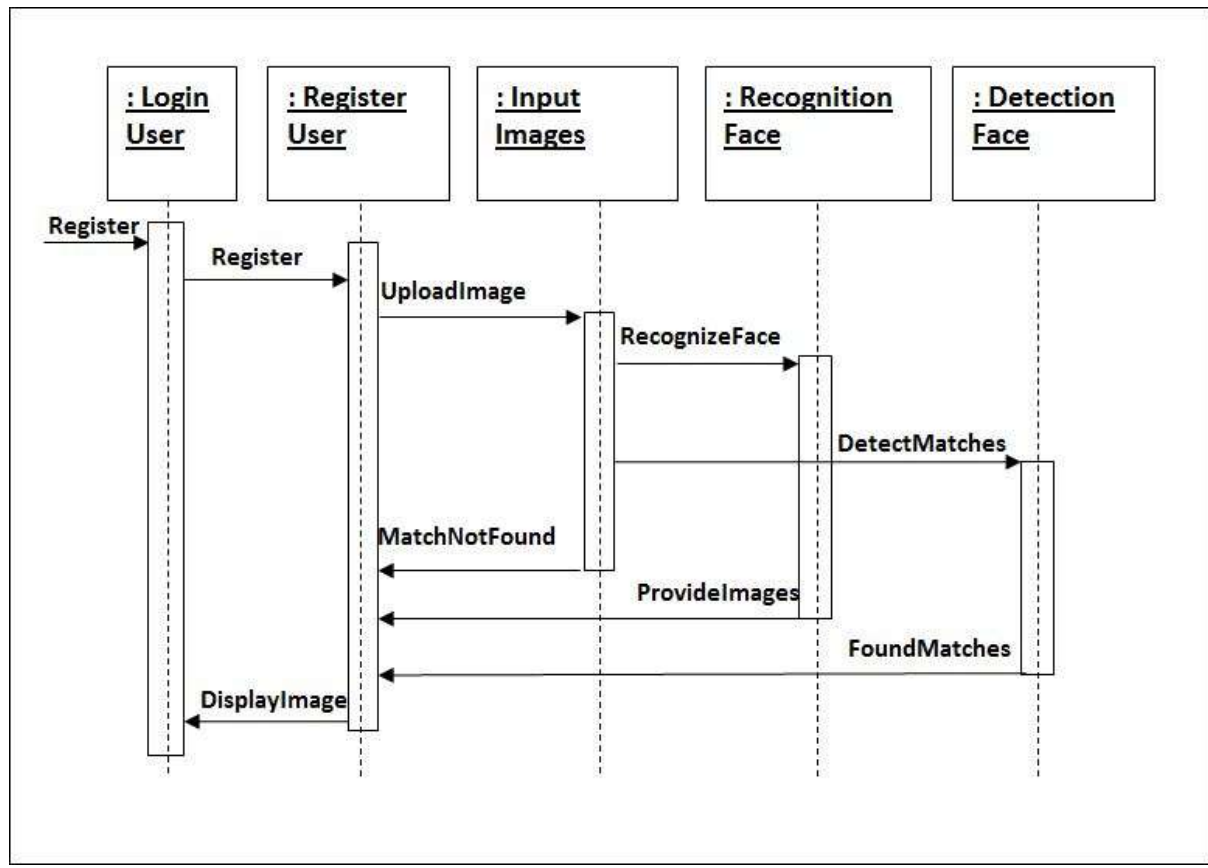


2-LEVEL DFD

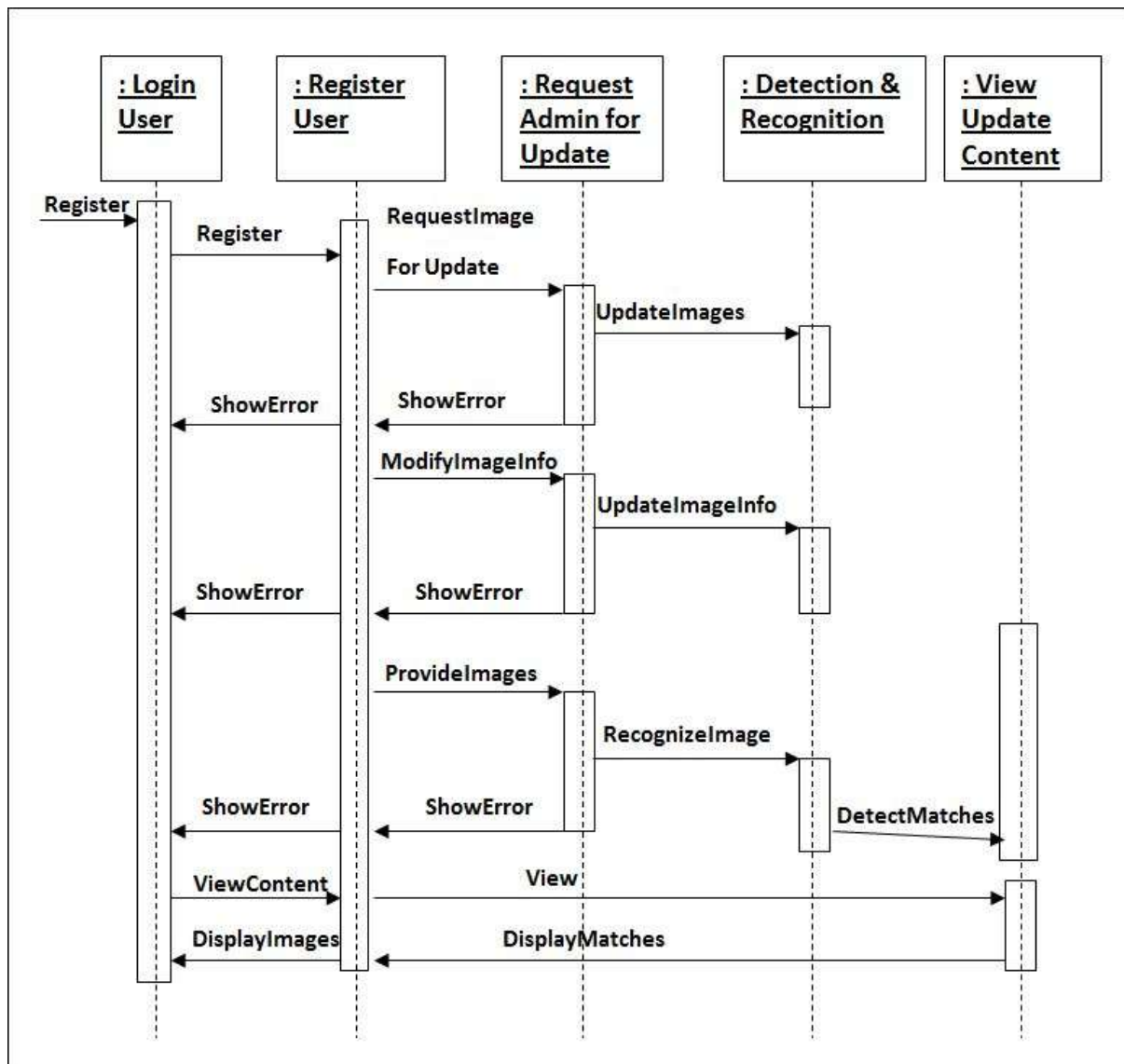


SEQUENCE DIAGRAMS

REGISTER IMAGES



IMAGES PROCESSING



ENTITY RELATIONSHIP MODEL

We will design a RDBMS for Face Recognition System. The entities and their attributes are listed below. Attributes in Bold letter is the unique key.

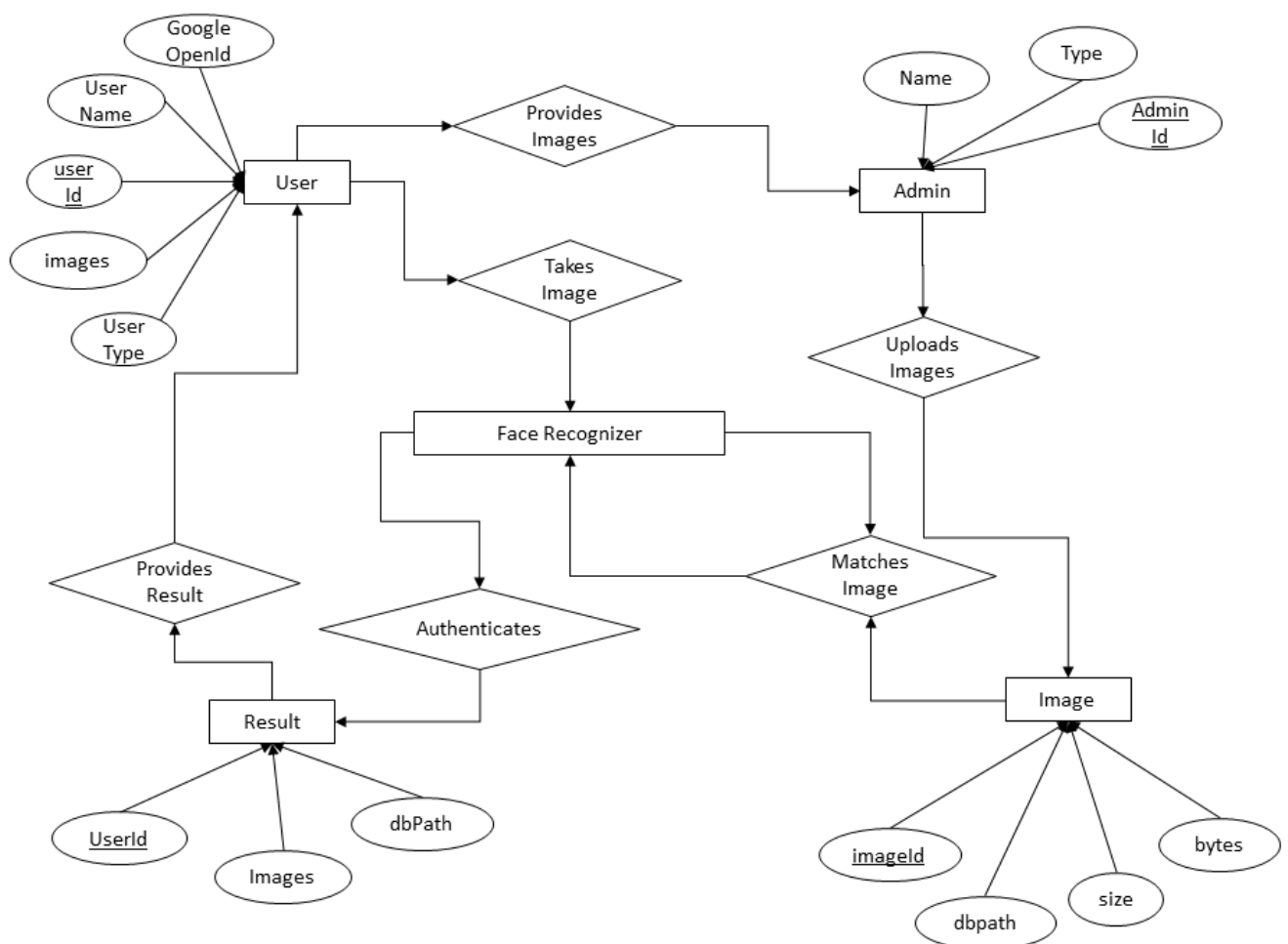
Entities	Attributes
User	userId , username, GoogleOpenId, type, images;
Admin	admin_ID , admin_Name, type
Image	imageId , dbpath, size, bytes;
Result	userId , images , dbpath

Relationship between Entities:

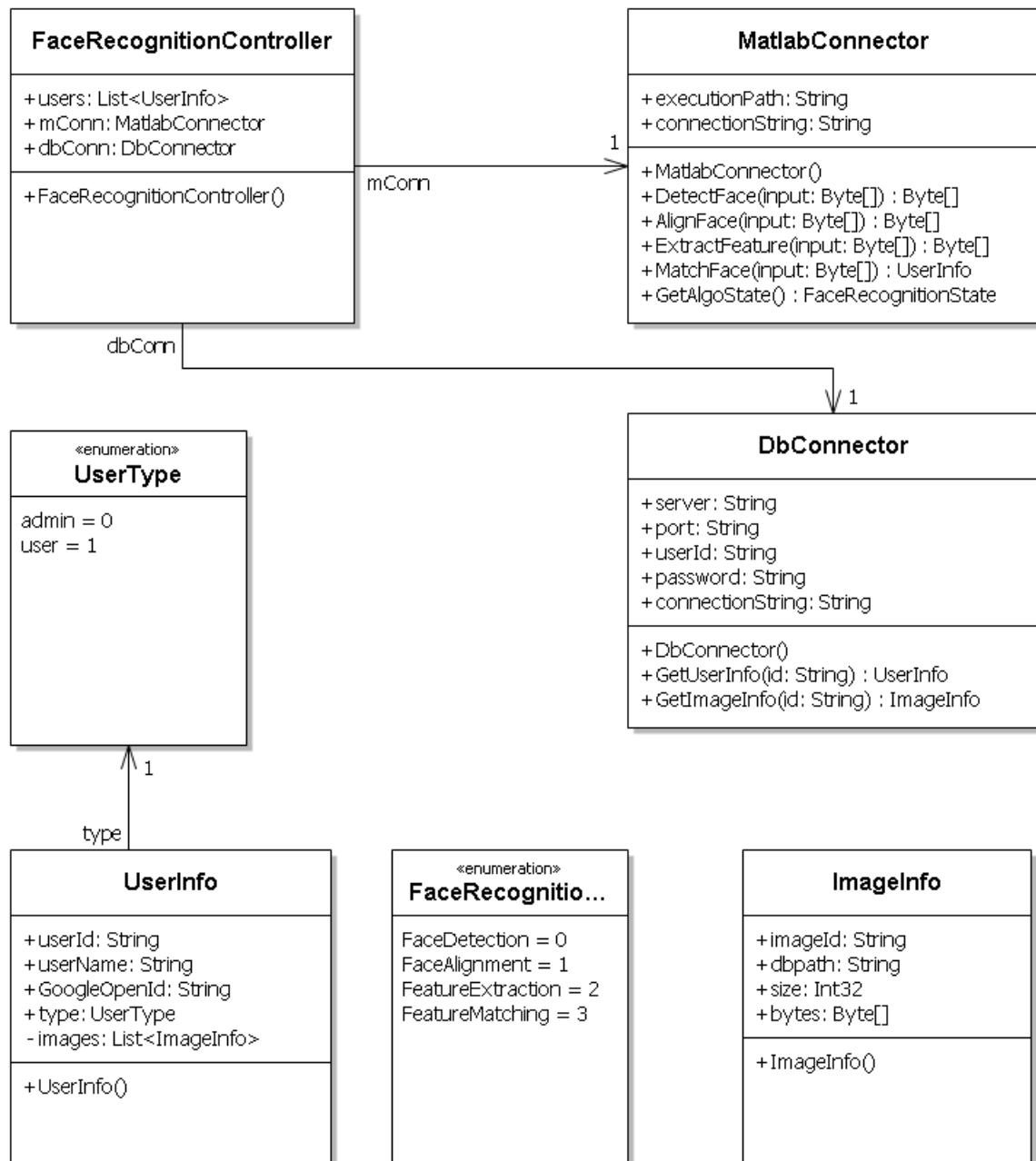
User hasimages1 : N

Admin uploads Images1 : N

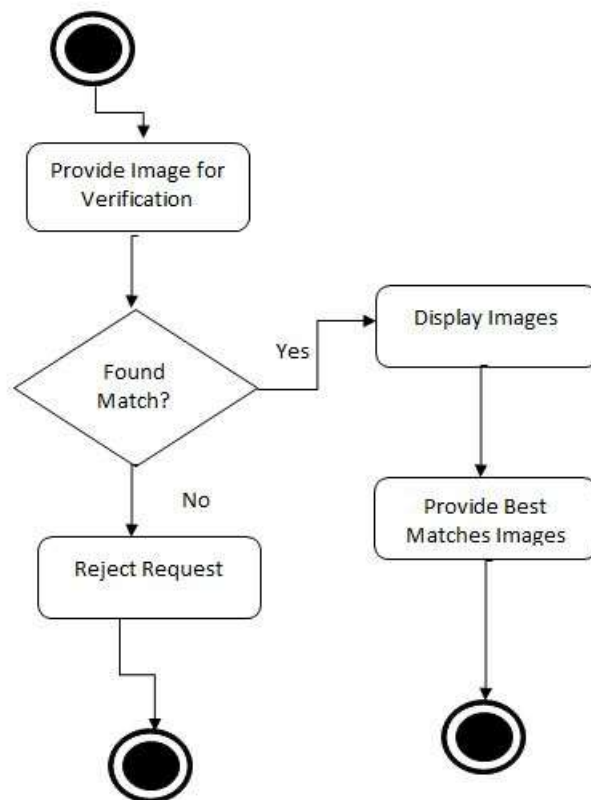
FaceRecognizer authenticates Users1 : N

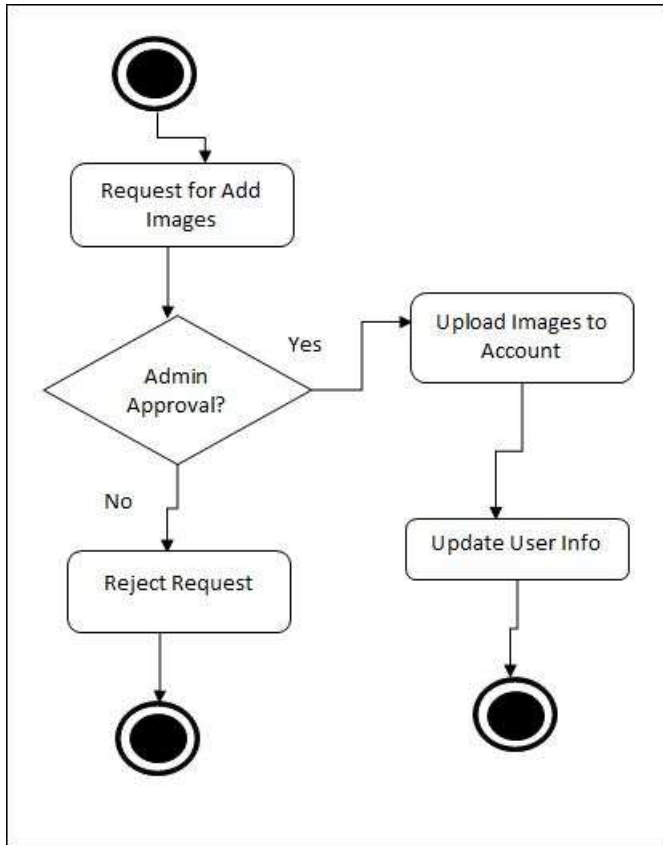


CLASS DIAGRAMS



ACTIVITY DIAGRAMS





SYSTEM DESIGN

MODULARISATION DETAILS




































Face Recognition System is divided into following main modules such as:

1. **Face Recognition System Windows Client:** This is the GUI for interacting with face Recognition System. This will be developed in WPF.
2. **Face Recognition System Controller:** This is the main controller of the whole system. We are following MVC architecture
3. **User Management system:** this module keeps track of the user of this software. It manages user privileges.
4. **Face Detection System:** Given a single image or video, an ideal face detector must be able to identify and locate all the present faces regardless of their position, scale, orientation, age, and expression.
5. **Face Alignment System:** In face alignment, facial components, such as eyes, nose, and mouth, and facial outline are located, and thereby the input face image is normalized in geometry and photometry.
6. **Feature Extraction Module:** Feature Extraction is done using Linear Subspace Approaches such as:
 - a. -Principal Component Analysis (PCA)
 - b. -Linear Discriminant Analysis (LDA)
 - c. -Independent Component Analysis (ICA)
7. **Feature Matching Handler:** The extracted features will be match with User input to determine a match.
8. **Matlab Connector:** Image processing algorithms will be implemented in MATLAB. This connector enables the bridge between .NET & Matlab code.
9. **Google Open Id Handler:** This will be Google app engine plugin which will use Google scripts to interact with Google account using face recognition system.
10. **Admin Controller:** This will be an admin panel where admin can manage users, provide access, modify data.
11. **Face Recognition System Database:** Face Recognition System will have a unified database for storing all the information. It can be a networked database or a database situated in the server machine

DATABASE & TABLE DETAILS

The database used for this software is called **frsdb**. A screenshot from the MySQL workbench is given below. It shows the tables and its columns. The first row is the primary key.

User:

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	userId	int(10)			No	None		      
<input type="checkbox"/>	username	varchar(15)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	GoogleOpenId	varchar(25)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	type	varchar(10)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	images	blob		BINARY	No	None		      

Admin:


















































	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	admin_ID	int(10)			No	None		      
<input type="checkbox"/>	admin_Name	varchar(50)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	type	varchar(10)	latin1_swedish_ci		No	None		      

Image:

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	imageId	int(10)			No	None		      
<input type="checkbox"/>	dbpath	varchar(200)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	size	int(10)			No	None		      
<input type="checkbox"/>	bytes	int(10)			No	None		      

Result






















	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	userId	int(10)			No	None		      
<input type="checkbox"/>	images	blob		BINARY	Yes	NULL		      
<input type="checkbox"/>	dbpath	varchar(200)	latin1_swedish_ci		No	None		      

Table	Column
User	userId, username, GoogleOpenId, type, images;
Admin	admin_ID, admin_Name, type
Image	imageId , dbpath, size, bytes;
Result	userId , images , dbpath

COMPLETE DATA STRUCTURE

DATA INTEGRITY AND CONSTRAINTS

We have used Integrity constraints in **FRS** to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. There are many types of integrity constraints in **FRS** that play a role in referential integrity.

Codd initially defined two sets of constraints but, in his second version of the relational model, he came up with four integrity constraints:

ENTITY INTEGRITY

In **FRS** we used various type of primary key and consciously we set the primary key property as not null. The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation. Having null value for the primary key implies that we cannot identify some tuples. This also specifies that there may not be any duplicate entries in primary key column key row.

REFERENTIAL INTEGRITY

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. It is a rule that maintains consistency among the rows of the two relations.

DOMAIN INTEGRITY

FRS has various type of data field with set by default value of Null because if the value is not provided by the user, the value will be set as null. The domain integrity states that every element from a relation should respect the type and restrictions of its corresponding attribute. A type can have a variable length which needs to be respected. Restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.

USER DEFINED INTEGRITY

A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business.

DATABASE DESIGN

The database used for this software is called frsdb. Database tables and corresponding keys are shown in tabular form. It shows the tables and its columns.

Screenshots of table structures:

TABLE: USER




































	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	userId	int(10)			No	None		      
<input type="checkbox"/>	username	varchar(15)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	GoogleOpenId	varchar(25)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	type	varchar(10)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	images	blob		BINARY	No	None		      

TABLE: ADMIN






















	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	admin_ID	int(10)			No	None		      
<input type="checkbox"/>	admin_Name	varchar(50)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	type	varchar(10)	latin1_swedish_ci		No	None		      

TABLE: IMAGE





























	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	imageId	int(10)			No	None		      
<input type="checkbox"/>	dbpath	varchar(200)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	size	int(10)			No	None		      
<input type="checkbox"/>	bytes	int(10)			No	None		      

TABLE: RESULT






















	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	userId	int(10)			No	None		      
<input type="checkbox"/>	images	blob		BINARY	Yes	NULL		      
<input type="checkbox"/>	dbpath	varchar(200)	latin1_swedish_ci		No	None		      

Table	Column
User	userId, username, GoogleOpenId, type, images;
Admin	admin_ID, admin_Name, type
Image	imageId , dbpath, size, bytes;
Result	userId , images , dbpath

USER INTERFACE DESIGN

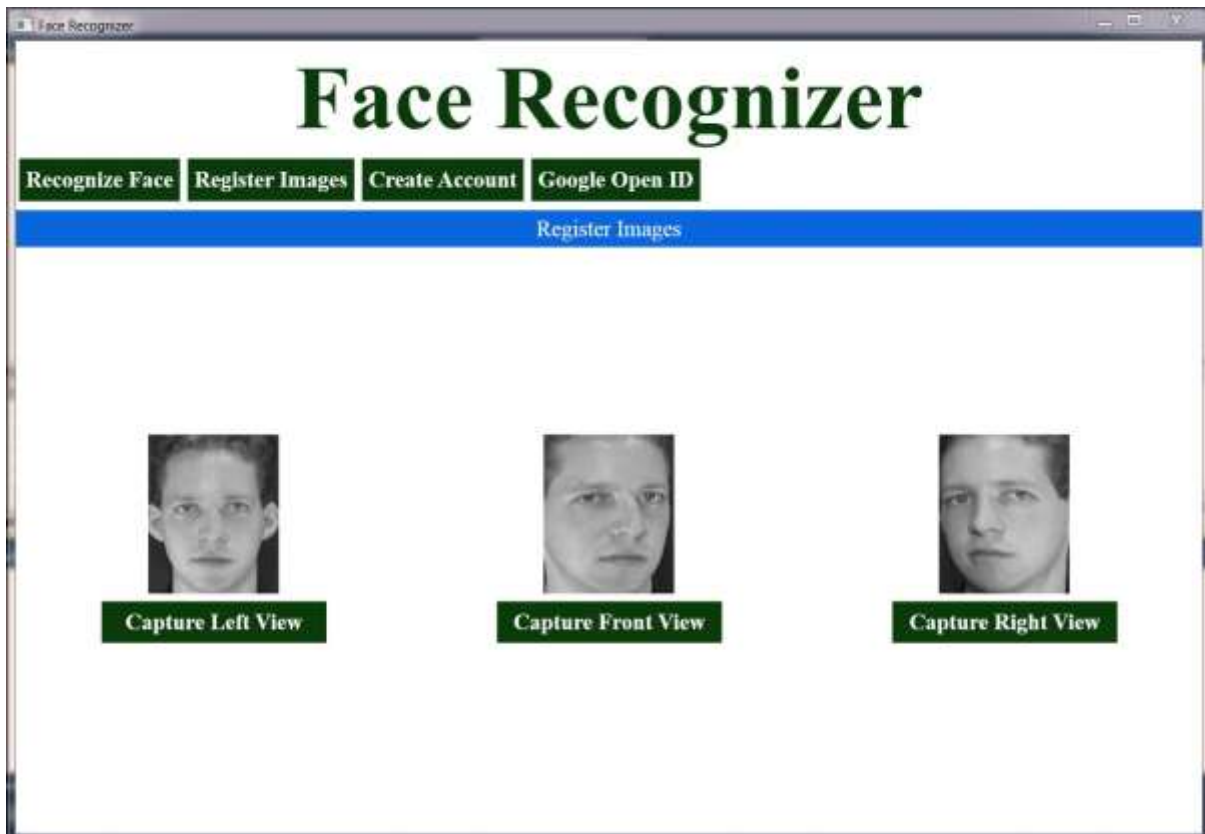
MAIN WINDOW



RECOGNIZE FACE



REGISTER IMAGE



CREATE ACCOUNT

The screenshot shows a web browser window titled "Face Recognizer". The main heading is "Face Recognizer" in a large, bold, green serif font. Below the heading is a navigation bar with four green buttons: "Recognize Face", "Register Images", "Create Account", and "Google Open ID". The "Create Account" button is highlighted with a blue background. Below the navigation bar is a blue header bar with the text "Create Account" in white. The main content area is white and contains a form for creating an account. On the left side of the form is a grayscale face image of a person. Below the image is a white input field and a green button labeled "Browse". To the right of the image are six text input fields labeled "Name", "Address", "Phone", "Email", "Occupation", and "Remark". At the bottom of the form are two green buttons: "Reset" and "Submit".

GOOGLE OPEN ID



TEST CASES (UNIT TEST CASES AND SYSTEM TEST CASES)

UNIT TEST CASES

Test Case Id	Type	Github ID	Subject	Test Name	Test Description	Step Name	Description	Expected Result
FRS-001	Manual	f3563be0a9c431104f52839039e86043cf640cf1	E:\DEVELOPERS_ZONE\GitHub\Face Recognition\code	Recognize Face in FRS	The purpose of this test is to verify that the Recognize Face system track the face properly.	Step 1	Click on Recognize Face from FRS main window.	Display Recognize Face window successfully.

FRS-002						Step 2	Click on Capture an image button to capture the target image.	FRS will capture the image.
FRS-003						Step 3	After capturing the image FRS check the image with its internal storage.	FRS will display the Recognition report.
FRS-004	M a n u a l	f3563be0a9c431104f52839039e86043cf644rt	E:\DEVELOPERS_ZONE\Git Hub\Face Recognition\code	Register images in FRS	The purpose of this test is to check that the Register images system work properly.	Step 1	Click on Register images from FRS main window.	Display Register images window successfully.
FRS-005						Step 2	Click on Capture Left View button to capture the left side View.	FRS will capture the left side View in its database.
FRS-006						Step 3	Click on Capture Front View button to capture the front side View.	FRS will capture the front side View in its database.
FRS-007						Step 4	Click on Capture Right View button to capture the Right side View.	FRS will capture the Right side View in its database.
FRS-008	M a n u	d01197ee4cd3bee9245874b5937ba740019fd131	E:\DEVELOPERS_ZONE\Git Hub\Face	Check Successful Create account for	The purpose of this test is to	Step 1	Click on Create Account from FRS.	Display Create Account window

	a 1		Recogniti on\code	FRS.	verify the creation new account work properly.			successfully.
FRS-009						Step 2	Click on submit button without putting proper information.	Display error message.
FRS-010						Step 3	Insert the details in Field like Name, Address, and Phone No. etc. And browse the currently taken image and click submit button.	Successful to Add account.
FRS-011						Step 4	Click on Browse button to get the image.	Successful to open a window for image.
FRS-012						Step 5	Put the image link in text field and add related data and click on.	Successful to create account.
FRS-013						Step 6	Insert the details in Field like Name, Address, and Phone No. etc. And browse the currently taken image and click reset button.	Successful to create account.

FRS-014	M a n u a l	d01197ee4c d3bee92458 74b5937ba7 40019f2893	E:\DEVELOPERS_ZONE\GitHub\Face Recognition\code	Check Successful login for google account for FRS .	The purpose of this test is to verify the login of google account work properly.	Step 1	Click on Google open Id from FRS.	Display Google open Id window successfully.
FRS-015						Step 2	Click on submit button without putting google user id and password.	Display error message.
FRS-016						Step 3	Insert the user id and password and click cancel button.	Successful to cancel login in.

SYSTEM TEST CASES

Test Case Id	Type	Github ID	Subject	Test Name	Test Description	Step Name	Description	Expected Result
FRS-017	M a n u a l	f3563be0a9 c431104f52 839039e860 43cf640cf1	E:\DEVELOPERS_ZONE\GitHub\Face Recognition\code	Recognize Face in FRS	The purpose of this test is to verify that the Recognize Face system track the face properly.	Step 1	Click on Recognize Face from FRS main window.	Display Recognize Face window successfully.
FRS-018						Step 2	Click on Capture an image button to capture the target image.	FRS will capture the image.

FRS-019						Step 3	After capturing the image FRS check the image with its internal storage.	FRS will display the Recognition report.
FRS-020	M a n u a l	f3563be0a9c431104f52839039e86043cf644rt	E:\DEVELOPERS_ZONE\Git Hub\Face Recognition\code	Register images in FRS	The purpose of this test is to check that the Register images system work properly.	Step 1	Click on Register images from FRS main window.	Display Register images window successfully.
FRS-021						Step 2	Click on Capture Left, Front, and Right View button to capture the Left, Front, and Right side View.	FRS will capture the Left, Front, and Right side View in its database.
FRS-022	M a n u a l	d01197ee4cd3bee9245874b5937ba740019fd131	E:\DEVELOPERS_ZONE\Git Hub\Face Recognition\code	Check Successful Create account for FRS .	The purpose of this test is to verify the creation new account work properly.	Step 1	Click on Create Account from FRS.	Display Create Account window successfully.
FRS-023						Step 2	Click on submit button without putting proper information.	Display error message.
FRS-024						Step 3	Insert the details in Field like Name, Address, and Phone No. etc. And browse the currently taken image and click submit	Successful to Add account.

							button.	
FRS-025						Step 4	Click on Browse button to get the image.	Successful to open a window for image.
FRS-026						Step 5	Insert the details in Field like Name, Address, and Phone No. etc. And browse the currently taken image and click reset button.	Successful to create account.
FRS-027	M a n u a l	d01197ee4c d3bee92458 74b5937ba7 40019f2893	E:\DEVELOPERS_ ZONE\Git Hub\Face Recognition\code	Check Successful login for google account for FRS .	The purpose of this test is to verify the login of google account work properly.	Step 1	Click on Google open Id from FRS.	Display Google open Id window successfully.
FRS-028						Step 2	Click on submit button without putting google user id and password.	Display error message.
FRS-029						Step 3	Insert the user id and password and click cancel button.	Successful to cancel login in.

CODING

COMPLETE PROJECT CODING

FRS GUI DESIGN CODING: FRSGUI

MainWindow.xaml

```
<Window x:Class="FaceRecognizer.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Face Recognizer" WindowState="Maximized" FontFamily="Times New Roman"
        FontSize="20">
    <Window.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="/FaceRecognizerStyle;component/ControlStyle.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Window.Resources>

    <DockPanel LastChildFill="True">

        <UniformGrid DockPanel.Dock="Top" Columns="1">
            <Label Name="lable1" VerticalContentAlignment="Center"
HorizontalContentAlignment="Center" FontFamily="Times New Roman" FontSize="80"
Foreground="#0B3B0B"
                Content="Face Recognizer" FontWeight="Bold">
            </Label>
        </UniformGrid>
        <UniformGrid DockPanel.Dock="Top" Rows="1" >

            </UniformGrid>
            <UniformGrid DockPanel.Dock="Top" Margin="2" Height="45">
                <TextBlock FontFamily="Sylfaen" FontWeight="Bold" FontStyle="Normal" >
                    <Button Name="recognizeFace" Style="{StaticResource ControlBtnStyle}"
Height="40" Click="recognizeFace_Click">Recognize Face</Button>
                    <Button Name="registerImages" Style="{StaticResource ControlBtnStyle}"
Height="40" Click="registerImages_Click">Register Images</Button>
                    <Button Name="createAccount" Style="{StaticResource ControlBtnStyle}"
Height="40" Click="createAccount_Click">Create Account</Button>
                    <Button Name="googleOpenID" Style="{StaticResource ControlBtnStyle}"
Height="40" Click="googleOpenID_Click">Google Open ID</Button>
                </TextBlock>
            </UniformGrid>
            <DockPanel DockPanel.Dock="Bottom" Name="infodocP" LastChildFill="True">
                <UniformGrid DockPanel.Dock="Top">
                    <UniformGrid.Background>
                        <ImageBrush
ImageSource="/FaceRecognizer;component/Images/face_recognition.jpg"></ImageBrush>
                    </UniformGrid.Background>
                </UniformGrid>
            </DockPanel>

        </DockPanel>

    </Window>
```

RecognizeFace.xaml

```
<UserControl x:Class="FaceRecognizer.RecognizeFace"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="403" d:DesignWidth="633">
    <UserControl.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="/FaceRecognizerStyle;component/ControlStyle.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </UserControl.Resources>

    <DockPanel >
        <Label DockPanel.Dock="Top" Background="#FF0966DF" Foreground="#FFFCFDFF"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Recognize
Face</Label>
        <DockPanel DockPanel.Dock="Top">

            <UniformGrid Name="manageUG" DockPanel.Dock="Top" Columns="2">

                <Grid>
                    <Image Name="capturePhoto" Stretch="Uniform" Height="142"
Margin="14,6,14,53" Width="203"
Source="/FaceRecognizer;component/Images/1.jpg"/></Image>
                    <Button Name="captureImage" Style="{StaticResource
ControlBtnStyle}" Margin="57,242,56,96" Height="39" Width="203">Capture an
Image</Button>
                </Grid>

                <Grid>
                    <Label Margin="57,154,57,8" Height="39" Width="202">Recognition
Report</Label>
                    <Image Name="capturedImgPhoto" Stretch="Uniform" Height="142"
Margin="14,6,14,53" Width="203"
Source="/FaceRecognizer;component/Images/2.jpg"/></Image>
                    <Label Margin="57,49,58,289" Height="39" Width="202">Nearest
Match</Label>
                    <UniformGrid Columns="2" Height="102" Width="310"
Margin="0,269,6,6">
                        <Label>Percentage Match</Label>
                        <Label>98%</Label>
                        <Label>Registered Name</Label>
                        <Label>Alan</Label>
                        <Label>Security Code</Label>
                        <Label>Alan2015</Label>
                    </UniformGrid>
                </Grid>

            </UniformGrid>
        </DockPanel>
    </DockPanel>
</UserControl>
```

```

    </DockPanel>
</UserControl>

```

CreateAccount.xaml

```

<UserControl x:Class="FaceRecognizer.CreateAccount"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="317" d:DesignWidth="592">
    <UserControl.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="/FaceRecognizerStyle;component/ControlStyle.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </UserControl.Resources>

    <DockPanel LastChildFill="True" >
        <Label DockPanel.Dock="Top" Background="#FF0966DF" Foreground="#FFFCFDFF"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Create
Account</Label>
        <DockPanel LastChildFill="True">
<UniformGrid DockPanel.Dock="Left" Columns="2">
            <Grid DockPanel.Dock="Left">
                <Image Name="prdctimgPhoto" Stretch="Uniform" Height="142"
Margin="14,6,14,53" Width="203"
Source="/FaceRecognizer;component/Images/2.jpg"></Image>
                <TextBox Name="prdctimagelinkTB" Margin="14,154,100,8" Width="117"
Height="39"></TextBox>
                <Button Name="prdctbrowseBtn" Style="{StaticResource ControlBtnStyle}"
Margin="137,154,6,8" Height="39" Width="88">Browse</Button>
            </Grid>

            <UniformGrid DockPanel.Dock="Right" Columns="2">
                <Label Style="{StaticResource LblStyle}" >Name</Label>
                <TextBox Name="nameTB"></TextBox>
                <Label Style="{StaticResource LblStyle}" >Address</Label>
                <TextBox Name="addressTB"></TextBox>
                <Label Style="{StaticResource LblStyle}" >Phone</Label>
                <TextBox Name="phoneTB"></TextBox>
                <Label Style="{StaticResource LblStyle}" >Email</Label>
                <TextBox Name="emailTB"></TextBox>
                <Label Style="{StaticResource LblStyle}" >Occupation</Label>
                <TextBox Name="occupationTB"></TextBox>
                <Label Style="{StaticResource LblStyle}" >Remark</Label>
                <TextBox Name="productdescriptionTB"></TextBox>
                <Button Name="resetBtn" Style="{StaticResource ControlBtnStyle}"
>Reset</Button>
                <Button Name="submitBtn" Style="{StaticResource
ControlBtnStyle}">Submit</Button>
            </UniformGrid>
        </DockPanel>
    </DockPanel>

```

```
</UserControl>
```

GoogleOpenID.xaml

```
<UserControl x:Class="FaceRecognizer.GoogleOpenID"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="151" d:DesignWidth="492">
    <UserControl.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="/FaceRecognizerStyle;component/ControlStyle.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </UserControl.Resources>

    <DockPanel LastChildFill="True" >
        <Label DockPanel.Dock="Top" Background="#FF0966DF" Foreground="#FFFCDFDF"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Google Open
ID</Label>
        <UniformGrid Name="upinfo" Background="White" DockPanel.Dock="Top"
Columns="5">
            <Label></Label>
            <Label></Label>
            <Label></Label>

            <DockPanel LastChildFill="True">
                <Label DockPanel.Dock="top"></Label>
                <Image DockPanel.Dock="top" Height="68" HorizontalAlignment="Left"
Name="image1" Stretch="Fill" VerticalAlignment="Top" Width="193"
Source="/FaceRecognizer;component/Images/wpids-photo-jan-14-2013-1115-am.jpg" />
                <UniformGrid Columns="2" DockPanel.Dock="Top">

                    <Label >User ID</Label>
                    <TextBox Name="idlinkTB"></TextBox>
                    <Label >Password</Label>
                    <PasswordBox Height="39"/>

                    <Button Name="cancelBtn" Style="{StaticResource ControlBtnStyle}"
>Cancel</Button>
                    <Button Name="loginBtn" Style="{StaticResource ControlBtnStyle}"
>Login</Button>

                </UniformGrid>
                <Label DockPanel.Dock="Bottom"></Label>
            </DockPanel>

        </UniformGrid>

    </DockPanel>

</UserControl>
```

RegisterImages.xaml

```
<UserControl x:Class="FaceRecognizer.RegisterImages"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="315" d:DesignWidth="764">
    <UserControl.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="/FaceRecognizerStyle;component/ControlStyle.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </UserControl.Resources>

    <DockPanel >
        <Label DockPanel.Dock="Top" Background="#FF0966DF" Foreground="#FFFCDF"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center">Register
Images</Label>
        <DockPanel DockPanel.Dock="Top">

            <UniformGrid Name="manageUG" DockPanel.Dock="Top" Columns="3">

                <Grid>

                    <Image Name="captureLeftViewPhoto" Stretch="Uniform"
Height="142" Margin="14,6,14,53" Width="203"
Source="/FaceRecognizer;component/Images/1.jpg"></Image>

                    <Button Name="captureLeftViewBtn" Style="{StaticResource
ControlBtnStyle}" Margin="57,242,56,96" Height="39" Width="203">Capture Left
View</Button>
                </Grid>

                <Grid>
                    <Image Name="captureFrontViewPhoto" Stretch="Uniform"
Height="142" Margin="14,6,14,53" Width="203"
Source="/FaceRecognizer;component/Images/2.jpg"></Image>
                    <Button Name="captureFrontViewBtn" Style="{StaticResource
ControlBtnStyle}" Margin="57,242,56,96" Height="39" Width="203">Capture Front
View</Button>
                </Grid>

                <Grid>
                    <Image Name="captureRightViewPhoto" Stretch="Uniform"
Height="142" Margin="14,6,14,53" Width="203"
Source="/FaceRecognizer;component/Images/4.jpg"></Image>
                    <Button Name="captureRightViewBtn" Style="{StaticResource
ControlBtnStyle}" Margin="57,242,56,96" Height="39" Width="203">Capture Right
View</Button>
                </Grid>

            </UniformGrid>
        </DockPanel>

    </DockPanel>
```



```
</UserControl>
```

GUI STYLE : FRSSYLES

ControlStyle.xaml

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <Style x:Key="ControlBtnStyle" TargetType="Button">
        <Setter Property="Background" Value="#0B3B0B" />
        <Setter Property="Foreground" Value="#FFFCFDFF" />
        <Setter Property="FontWeight" Value="Bold" />
        <Setter Property="BorderBrush" Value="#FF3DC43D" />
        <Setter Property="FontFamily" Value="Times New Roman" />
        <Setter Property="OpacityMask" Value="White" />
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="Button">
                    <Border Name="border"
                        BorderThickness="1"
                        Padding="6,2"
                        BorderBrush="White"
                        CornerRadius="0"
                        Background="{TemplateBinding Background}"
                        <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center" />
                    </Border>
                    <ControlTemplate.Triggers>
                        <Trigger Property="IsMouseOver" Value="True">
                            <Setter TargetName="border" Property="BorderBrush"
Value="#FFDFEBF2" />
                            <Setter Property="Button.Background" Value="#FFDFEBF2" />
                            <Setter Property="Button.Foreground" Value="#0B3B0B" />
                        </Trigger>
                        <Trigger Property="IsPressed" Value="True">
                            <Setter TargetName="border" Property="BorderBrush"
Value="White" />
                            <Setter Property="Button.Background" Value="#D8D8D8" />
                        </Trigger>

                        <!--<Trigger Property="IsSelected" Value="true">
                            <Setter Property="Fill" Value="Blue" />
                        </Trigger>-->

                        <!--<Trigger Property="ToggleButton.IsChecked" Value="true">
                            <Setter Property="Button.Background" Value="White" />
                        </Trigger>
                        <Trigger Property="IsEnabled" Value="True">
                            <Setter TargetName="Border" Property="Background"
Value="Red" />
                            <Setter TargetName="Border" Property="BorderBrush"
Value="Red" />
                            <Setter Property="Foreground" Value="Red"/>
                        </Trigger>-->
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
```

```

    </Setter>
</Style>

<Style x:Key="TItemStyle" TargetType="TabItem">
    <Setter Property="FontFamily" Value="Times New Roman"/>
    <Setter Property="FontSize" Value="20"/>
    <Setter Property="Foreground" Value="#FFFCDF"/>

    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type TabItem}">
                <Grid>
                    <Border Name="Border" Margin="0,0,0,0" Background="#FF0966DF"
                        BorderThickness="1,1,1,1" CornerRadius="0">
                        <ContentPresenter x:Name="ContentSite"
VerticalAlignment="Center"
                            HorizontalAlignment="Center"
                            ContentSource="Header" Margin="12,2,12,2"
                            RecognizesAccessKey="True">
                            <ContentPresenter.LayoutTransform>
                                <RotateTransform Angle="0" />
                            </ContentPresenter.LayoutTransform>
                        </ContentPresenter>
                    </Border>
                </Grid>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsSelected" Value="True">
                        <Setter TargetName="Border" Property="Background"
Value="#FFDFEBF2" />
                        <Setter Property="Button.Foreground" Value="#FF0966DF" />
                    </Trigger>
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter TargetName="Border" Property="Background"
Value="#FFDFEBF2" />
                        <Setter Property="Button.Foreground" Value="#FF0966DF" />
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
    <Setter Property="HeaderTemplate">
        <Setter.Value>
            <DataTemplate>
                <Border x:Name="grid" >
                    <ContentPresenter>
                        <ContentPresenter.Content>
                            <TextBlock Margin="4" FontSize="20"
Text="{TemplateBinding Content}"/>
                        </ContentPresenter.Content>
                    </ContentPresenter>
                </Border>

                <!--<DataTemplate.Triggers>
                    <DataTrigger Binding="{Binding RelativeSource={RelativeSource
Mode=FindAncestor,AncestorType={x:Type TabItem}},Path=IsSelected}" Value="True">
                        <Setter TargetName="grid" Property="Background"
Value="Green"/>
                    </DataTrigger>
                </DataTemplate.Triggers-->
            </DataTemplate>
        </Setter.Value>
    </Setter>

```

```

        </Setter>
    </Style>

    <Style x:Key="CheckBtype" TargetType="CheckBox">
        <Setter Property="Width" Value="80" />
        <Setter Property="Background" Value="#FFF9F9EC" />
        <Setter Property="BorderBrush" Value="#FFC4C458" />
        <Setter Property="Foreground" Value="#000000" />
    </Style>

    <Style x:Key="pBstyle" TargetType="PasswordBox">
        <Setter Property="Control.Template" >
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type PasswordBox}">
                    <Border x:Name="border" BorderBrush="{TemplateBinding
BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}"
Background="{TemplateBinding Background}" SnapsToDevicePixels="True">
                        <Grid>
                            <ScrollViewer x:Name="PART_ContentHost" Focusable="False"
HorizontalScrollBarVisibility="Hidden" VerticalScrollBarVisibility="Hidden"/>

                            <TextBlock x:Name="InternalWatermarkLabel"
Text="{TemplateBinding Tag}"
Visibility="Collapsed" Focusable="False"
VerticalAlignment="Top" Margin=" 5 1 0 0"
Foreground="Silver"
Background="Transparent"/>
                        </Grid>
                    </Border>
                    <ControlTemplate.Triggers>
                        <MultiTrigger>
                            <MultiTrigger.Conditions>
                                <Condition Property="IsFocused" Value="False" />
                                <!--<Condition Property="Password" Value="c" />-->
                            </MultiTrigger.Conditions>
                            <MultiTrigger.Setters>
                                <Setter Property="Visibility"
TargetName="InternalWatermarkLabel"
Value="Visible" />
                            </MultiTrigger.Setters>
                        </MultiTrigger>
                        <Trigger Property="IsEnabled" Value="False">
                            <Setter Property="Opacity" TargetName="border"
Value="0.56"/>
                        </Trigger>
                        <Trigger Property="IsMouseOver" Value="True">
                            <Setter Property="BorderBrush" TargetName="border"
Value="#FF7EB4EA"/>
                        </Trigger>
                        <Trigger Property="IsKeyboardFocused" Value="True">
                            <Setter Property="BorderBrush" TargetName="border"
Value="#FF569DE5"/>
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>

    <Style x:Key="TitlebarBTN" TargetType="Button">
        <Setter Property="OverridesDefaultStyle" Value="True"/>
        <Setter Property="Margin" Value="0"/>

```

```

        <Setter Property="FontWeight" Value="Bold" />
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="Button">
                    <Border Name="border"
                        BorderThickness="1"
                        Padding="4,2"
                        BorderBrush="White"
                        CornerRadius="0"
                        Background="{TemplateBinding Background}">
                        <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center" />
                    </Border>
                    <ControlTemplate.Triggers>
                        <Trigger Property="IsMouseOver" Value="True">
                            <Setter TargetName="border" Property="BorderBrush"
Value="White" />
                            <Setter Property="Button.Background" Value="#D8D8D8" />
                        </Trigger>
                        <Trigger Property="IsPressed" Value="True">
                            <Setter TargetName="border" Property="BorderBrush"
Value="White" />
                            <Setter Property="Button.Background" Value="#2E9AFE" />
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>

    <Style x:Key="{x:Type ToolTip}" TargetType="{x:Type ToolTip}">
        <Setter Property="Background" Value="Black"/>
        <Setter Property="BorderBrush" Value="#D8D8D8"/>
        <Setter Property="Foreground" Value="White" />
        <Setter Property="Padding" Value="7" />
        <Setter Property="FontWeight" Value="Bold" />

        <!--<Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="ToolTip">
                <Canvas Width="200" Height="100">
                    <Path x:Name="Container"
                        Canvas.Left="0"
                        Canvas.Top="0"
                        Margin="20"
                        Data="M 0,40 L15,50 15,80 150,80 150,0 15,0 15,30"
                        Fill="{TemplateBinding Background}"
                        Stroke="Black">
                    <Path.Effect>
                        <DropShadowEffect BlurRadius="10"
                            Opacity="0.5"
                            ShadowDepth="4" />
                    </Path.Effect>
                </Path>
                <TextBlock Canvas.Left="50"
                    Canvas.Top="28"
                    Width="100"
                    Height="65"
                    Text="{TemplateBinding Content}"
                    TextWrapping="Wrapwithoverflow" />
            </Canvas>
        </ControlTemplate>
        -->
    </Style>

```

```

        </Setter.Value>
    </Setter>-->
</Style>

<Style x:Key="commonTBtype" TargetType="TextBox">

    <Setter Property="Control.Template" >
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type TextBox}">
                <Border x:Name="border" BorderBrush="{TemplateBinding
BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}"
Background="{TemplateBinding Background}" SnapsToDevicePixels="True">
                    <Grid>
                        <ScrollViewer x:Name="PART_ContentHost" Focusable="False"
HorizontalScrollBarVisibility="Hidden" VerticalScrollBarVisibility="Hidden"/>

                        <TextBlock x:Name="InternalWatermarkLabel"
Text="{TemplateBinding Tag}"
Visibility="Collapsed" Focusable="False"
VerticalAlignment="Top" Margin=" 5 1 0 0"
Foreground="Silver"
Background="Transparent"/>
                    </Grid>
                </Border>
                <ControlTemplate.Triggers>
                    <MultiTrigger>
                        <MultiTrigger.Conditions>
                            <Condition Property="IsFocused" Value="False" />
                            <Condition Property="Text" Value="" />
                        </MultiTrigger.Conditions>
                        <MultiTrigger.Setters>
                            <Setter Property="Visibility"
TargetName="InternalWatermarkLabel"
Value="Visible" />
                        </MultiTrigger.Setters>
                    </MultiTrigger>
                    <Trigger Property="IsEnabled" Value="False">
                        <Setter Property="Opacity" TargetName="border"
Value="0.56"/>
                    </Trigger>
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter Property="BorderBrush" TargetName="border"
Value="#FF7EB4EA"/>
                    </Trigger>
                    <Trigger Property="IsKeyboardFocused" Value="True">
                        <Setter Property="BorderBrush" TargetName="border"
Value="#FF569DE5"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>

</Style>

<Style x:Key="LblStyle" TargetType="Label">
    <Setter Property="Background" Value="Transparent" />
    <!--<Setter Property="Foreground" Value="#FF0966DF" />-->
    <!--<Setter Property="FontWeight" Value="Bold" />-->
    <Setter Property="Margin" Value="0"/>
    <Setter Property="Padding" Value="7" />

```

```

</Style>

<Style x:Key="TextBlockStyle" TargetType="TextBlock">
    <Setter Property="Background" Value="Transparent" />
    <!--<Setter Property="FontWeight" Value="Bold" />-->
    <Setter Property="Margin" Value="20"/>
    <Setter Property="Padding" Value="7" />
    <Setter Property="TextWrapping" Value="Wrap" />
</Style>

<Style x:Key="HyperlinkStyle" TargetType="Hyperlink">
    <Setter Property="FontSize" Value="20"/>
    <Setter Property="FontFamily" Value="Times New Roman"/>

    <Setter Property="Foreground" Value="Goldenrod"/>
    <Style.Triggers>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Foreground" Value="White"/>
        </Trigger>
    </Style.Triggers>
</Style>

<Style x:Key="UserCntrlStyle" TargetType="UserControl">
    <Setter Property="FontFamily" Value="Times New Roman"/>
    <Setter Property="FontSize" Value="20"/>
    <Setter Property="Foreground" Value="#FF318931"/>
</Style>

<Style x:Key="comboboxStyle" TargetType="ComboBox">
    <Setter Property="Foreground" Value="Black" />
    <Setter Property="IsEditable" Value="True" />
    <Setter Property="VerticalContentAlignment" Value="Center" />
    <Setter Property="HorizontalContentAlignment" Value="Left" />
</Style>
</ResourceDictionary>

```

MatlabConnector

```

publicclassMatlabConnector
{
publicstringexecutionPath;
publicstringconnectionString;

```

```

public byte[] DetectFace(byte[] input)

    {
    throw new NotImplementedException();
    }
public byte[] AlignFace(byte[] input)
    {
    throw new NotImplementedException();
    }
public byte[] ExtractFeature(byte[] input)
    {
    throw new NotImplementedException();
    }
public UserInfo MatchFace(byte[] input)
    {
    throw new NotImplementedException();
    }
public FaceRecognitionState GetAlgoState()
    {
    throw new NotImplementedException();
    }
}

```

FaceRecognitionController

```

public class FaceRecognitionController
    {
    public List<UserInfo> users;
    public MatlabConnector mConn;
    public DbConnector dbConn;
    }

```

BUSINESS LOGIC – MATLAB

nonlinear_subspace_projection_PhD.m

```
function feat = nonlinear_subspace_projection_PhD(X, model);

%% Init
feat=[];

%% Check inputs

%check number of inputs
if nargin <2
    disp('Wrong number of input parameters! The function requires two input arguments.')
    return;
elseif nargin >2
    disp('Wrong number of input parameters! The function requires two input arguments.')
    return;
elseif nargin==2
    %get size of testing data
    [a,b]=size(X);

    %check model
    if isfield(model, 'W')~=1
        disp('There is no subspace basis defined. Missing model.W!')
        return;
    end

    if isfield(model, 'dim')~=1
        disp('There is no subspace dimensionality defined. Missing model.dim!')
        return;
    end

    if isfield(model, 'eigs')~=1
        disp('There are no eigenvalues associated with the subspace basis. Missing model.eigs!')
        return;
    end

    if isfield(model, 'train')~=1
        disp('There are no training features defined. Missing model.train!')
        return;
    end

    if isfield(model, 'J')~=1
        disp('There is no auxiliary matrix defined. Missing model.J!')
```



```

        return;
    end

    if isfield(model, 'K') ~= 1
        disp('There is no training kernel matrix defined. Missing
model.K!')
        return;
    end

    if isfield(model, 'typ') ~= 1
        disp('There is no kernel type defined. Missing model.typ!')
        return;
    end

    if isfield(model, 'args') ~= 1
        disp('There are no kernel arguments defined. Missing model.args!')
        return;
    end

    %check that the test and trainnig data are of the same dimensionality
    [c,d]=size(model.X);
    if c~=a
        disp('The dimensionality of the training and test data must be the
same.')
        return;
    end
end

%% Print info to command prompt and init. operations
[c,d]=size(model.J);

%we assume that the data is contained in the columns
% disp(sprintf('The training data comprises %i samples (images) with %i
variables (pixels).', b, a))
% disp('If this should be the other way around, please transpose the
training-data matrix.')

%% Compute the test features

%compute the test kernel matrix
K = compute_kernel_matrix_PhD(X,model.X,model.typ,model.args);

%center the test kernel matrix
Jt=ones(b,1)/c;
J=ones(c,1);
K=K-((Jt*J')*model.K)-(K*(1/c)*(J*J'))+(Jt*J'*model.K*(1/c)*(J*J'));

%need to transpose the result - because of kernel matrix comptation
K=K';

%feature computation
feat = model.W'*K;

```

perform_kfa_PhD.m

```
function model = perform_kfa_PhD(X, ids, kernel_type, kernel_args,n);

%% Init
model = [];

%% Check inputs

%check number of inputs
if nargin <2
    disp('Wrong number of input parameters! The function requires at least
two input arguments.')
    return;
elseif nargin >5
    disp('Wrong number of input parameters! The function takes no more than
five input arguments.')
    return;
elseif nargin==2
    [a,b]=size(unique(ids));
    n = max([a,b])-1;
    kernel_type = 'poly'
    kernel_args = [0 2];
elseif nargin==3
    [a,b]=size(unique(ids));
    n = max([a,b])-1;
    if strcmp(kernel_type,'poly')==1
        kernel_args = [0 2];
    elseif strcmp(kernel_type,'fpp')==1
        kernel_args = [0 .8];
    elseif strcmp(kernel_type,'tanh')==1
        kernel_args = 0;
    else
        disp('The entered kernel type was not recognized as a supported
kernel type.')
        return;
    end
elseif nargin==4
    [a,b]=size(unique(ids));
    n = max([a,b])-1;
end

%check if ids is a vector
if isvector(ids)==0
    disp('The second parameter "ids" needs to be a vector.')
    return;
end

% check if n is not to big
[a,b]=size(unique(ids));
if n>max([a,b])-1;
    disp('The parameter "n" must not be larger than the number of classes
minus one. Decreasing "n"!')
    n = max([a,b])-1;
end

%check that each image in X has a class label
[a,b]=size(X);
if b~=length(ids)
```

```

        disp('The label vector "ids" needs to be the same size as the number of
samples in X.')
        return;
    end

%we assume that the data is contained in the columns - print to prompt
disp(sprintf('The training data comprises %i samples (images) with %i
variables (pixels).', b, a))
disp('If this should be the other way around, please transpose the
training-data matrix.')

%% Compute the KFA subspace - main part

%compute the training data kernel matrix
K = compute_kernel_matrix_PhD(X,X, kernel_type, kernel_args);
model.K = K;

%center kernel
J = ones(b,b)/b;
Kc = K - J*K - K*J + J*K*J;
model.J = J;

%compute W using auxilary function
W=zeros(b,b);
W=return_W(W,ids);

%construct eigenproblem using Tickhonov regularization
epsi = 1e-10*min(min(Kc*Kc)); %some small regularization constant
Crit = (Kc*Kc+epsi*eye(b,b))\ (Kc*W*Kc);
clear W

%solve eigenproblem
[U,V,L]=svd(Crit);
clear L Crit
Alpha = normc(U(:,1:n));
clear U

%normalize Alpha to be unit length in F
R=Alpha'*Kc*Alpha;
norms = real(diag(R));
for i=1:n
    Alpha(:,i)=Alpha(:,i)/sqrt(norms(i));
end

%construct some outputs
model.W = Alpha;
model.dim = n;
model.eigs = diag(V);
model.typ = kernel_type;
model.args = kernel_args;
model.X = X;

%% Construct training features
model.train = Alpha'*Kc;

```

```

%% This is the auxiliary function used to produce W
% for within-class scatter comutation
function W=return_W(W,ids)

id_unique = unique(ids);
[c,d]=size(id_unique);
num_of_class = max([c,d]); %this is the number of classes

for i=1:num_of_class
    [dummy,ind]=find(id_unique(i)==ids);
    [x,y] = meshgrid(ind,ind);
    elem_val = 1/sum(length(ind));
    W(x,y)=elem_val;
end

```

nn_classification_PhD.m

```

function results = nn_classification_PhD(train, train_ids, test, test_ids,
n, dist, match_kind);

%% Init
results = [];

%% Check inputs

%check number of inputs
if nargin <4
    disp('Wrong number of input parameters! The function requires at least
four input arguments.')
    return;
elseif nargin >7
    disp('Wrong number of input parameters! The function takes no more than
seven input arguments.')
    return;
elseif nargin==4
    n = size(train,1);
    dist = 'cos';
    match_kind = 'all'
elseif nargin ==5
    dist = 'cos';
    match_kind = 'all';
elseif nargin ==6
    match_kind = 'all';

```

```

end

%check distance
if ischar(dist)~=1
    disp('The parameter "dist" needs to be a string - a valid one!')
    return;
end

%check if test and train features are of same size
[a,b]=size(train);
[c,d]=size(test);
if a~=c
    disp('The number of features in the training and test vectors needs to
be identical.')
    return;
end

%check the matching type
if ischar(match_kind)~=1
    disp('The parameter "match_kind" has to be a string - a valid one!')
    return;
end

%check if match_kind is a valid string
if strcmp(match_kind,'all')==1 || strcmp(match_kind,'sep')==1 ||
strcmp(match_kind,'allID')==1
    %ok
else
    disp('The input string "match_kind" is not a valid identifier:
all|sep.')
    return;
end

%check if n is not to big
if n>a
    disp('The parameter "n" cannot be larger than the dimensionality of
your feature vectors. Decreasing "n" to maximal allowed size.')
    n=a;
end

%check if the distance is valid
if strcmp(dist,'cos')==1 || strcmp(dist,'euc')==1 || strcmp(dist,'ctb')==1
|| strcmp(dist,'mahcos')==1
    %ok
else
    disp('The parameter "dist" need to be a valid string identifier: cos,
euc, ctb, or mah. Switching to deafults (cos).')
    dist = 'cos';
end

%check if ids (class labels) are vectors
if isvector(train_ids)==0
    disp('The second parameter "train_ids" needs to be a vector of numeric
values.')
    return;
end

if isvector(test_ids)==0
    disp('The second parameter "train_ids" needs to be a vector of numeric

```

```

values.')
    return;
end

%check that each feature vector has a label
[a,b]=size(train);
if b~=length(train_ids)
    disp('The label vector "train_ids" needs to be the same size as the
number of samples in "train".')
    return;
end

[c,d]=size(test);
if d~=length(test_ids)
    disp('The label vector "test_ids" needs to be the same size as the
number of samples in "test".')
    return;
end

%% Prepare data and do the matching - main part

%get sample sizes
[a,b]=size(train);
[c,d]=size(test);

%precompute the inverse of the covariance if needed
covar = inv(cov(train(1:n,:),')));

%select type of matching
if strcmp(match_kind,'sep')==1
    %the "sep" option separates the test data into two groups - clients and
    %impostors (needed by some protocols) - this is useful for verification
    %with an unseen impostor set (e.g., XM2VTS), the identification score
    %can still be computed from the client similarity matrix

    %write matching mode to result structure
    results.mode = 'sep';

    %find impostor labels
    impostor_ids = setdiff(test_ids,train_ids);
    if size(impostor_ids,1)==0 || size(impostor_ids,2)==0
        disp('In separation mode, the query data matrix has to feature at
least some subjects that are not in the target set.')
        return;
    end

    %find client labels
    client_ids = intersect(test_ids,train_ids);
    if size(client_ids,1)==0 || size(client_ids,2)==0
        disp('In separation mode, the query data matrix has to feature at
least some subjects that are also in the target set.')
        return;
    end

    %some reporting
    disp('Entering separation mode!')

```

```

%seperate and compute the distances

%clients - extract client data
num_of_experiments = 0;
for i=1:length(client_ids)
    [incr,dummy] = find(client_ids(i)==test_ids);
    num_of_experiments = num_of_experiments + sum(incr);
end

%we rather add another for loop now that we know the size - this is
%faster than reallocating space in each iteration - Štrikam
test_cli = zeros(n,num_of_experiments);
test_cli_ids = zeros(1,num_of_experiments);
cont=1;
for i=1:length(client_ids)
    [incr,dummy] = find(client_ids(i)==test_ids);
    test_cli(:,cont:cont+length(incr)-1) = test(1:n,dummy);
    test_cli_ids(1,cont:cont+length(incr)-1) = client_ids(i);
    cont = cont+length(incr);
end

%some reporting
disp('Computing client similarity matrix ...')

%compute distances
results.client_dist = zeros(num_of_experiments,b);
results.same_cli_id = zeros(num_of_experiments,b);
for i=1:num_of_experiments
    for j=1:b
        results.client_dist(i,j) =
return_PhD_distance(train(1:n,j),test_cli(1:n,i),dist,covar);
        if train_ids(j)==test_cli_ids(i)
            results.same_cli_id(i,j)=1;
        else
            results.same_cli_id(i,j)=0;
        end
    end
end
results.client_horizontal_ids = train_ids;
results.client_vertical_ids = test_cli_ids;

disp('Done.')

disp('Computing impostor scores ...')
%impostors - extract impostor data
num_of_experiments = 0;
for i=1:length(impostor_ids)
    [incr,dummy] = find(impostor_ids(i)==test_ids);
    num_of_experiments = num_of_experiments + sum(incr);
end

%we rather add another for loop now that we know the size - this is
%faster than reallocating space in each iteration - Štrikam
test_imp = zeros(n,num of experiments);

```

```

test_imp_ids = zeros(1,num_of_experiments);
cont=1;
for i=1:length(impostor_ids)
    [incr,dummy] = find(impostor_ids(i)==test_ids);
    test_imp(:,cont:cont+length(incr)-1) = test(1:n,dummy);
    test_imp_ids(1,cont:cont+length(incr)-1) = impostor_ids(i);
    cont = cont+length(incr);
end

%compute distances
results.imp_dist = zeros(num_of_experiments,b);
results.same_imp_id = zeros(num_of_experiments,b);
for i=1:num_of_experiments
    for j=1:b
        results.imp_dist(i,j) =
return_PhD_distance(train(1:n,j),test_imp(1:n,i),dist,covar);
        if train_ids(j)==test_imp_ids(i)
            results.same_imp_id(i,j)=1;
        else
            results.same_imp_id(i,j)=0;
        end
    end
end
results.imp_horizontal_ids = train_ids;
results.imp_vertical_ids = test_imp_ids;
results.dist = dist;
results.dim = n;
disp('Done.')
elseif strcmp(match_kind,'all')==1
    %the "all" option matches each samples from the test matrix against all
    %samples form the train matrix

    %write matching mode to result structure
    results.mode = 'all';

    %find client labels
    client_ids = intersect(test_ids,train_ids);
    if size(client_ids,1)==0 || size(client_ids,2)==0
        disp('In all mode, the query data matrix has to feature at least
some subjects that are also in the target set.')
        return;
    end

    %some reporting
    disp('Entering all mode!')
    disp('Computing similarity matrix ...')

    %compute distances
    results.match_dist = zeros(d,b);
    %results.same_id = zeros(d,b);
    for i=1:d
        for j=1:b
            results.match_dist(i,j) =
return_PhD_distance(train(1:n,j),test(1:n,i),dist,covar);
            if train_ids(j)==test_ids(i)
                results.same_cli_id(i,j)=1;
            else
                results.same_cli_id(i,j)=0;
            end
        end
    end
end

```



```

        end
    end
    results.horizontal_ids = train_ids;
    results.vertical_ids = test_ids;
    results.dist = dist;
    results.dim = n;
    disp('Done.')
elseif strcmp(match_kind, 'allID')==1
    %the "allID" option matches each samples from the test matrix against
    all
    %IDs form the train matrix - each ID here is represented with the mean
    %feature vector of that ID - this is just for me and I will not
    %document it

    %write matching mode to result structure
    results.mode = 'all';

    %find client labels
    client_ids = intersect(test_ids, train_ids);
    if size(client_ids,1)==0 || size(client_ids,2)==0
        disp('In all mode, the query data matrix has to feature at least
some subjects that are also in the target set.')
        return;
    end

%
%     [incr,dummy] = find(impostor_ids(i)==test_ids);
%         test_imp(:,cont:cont+length(incr)-1) = test(1:n,dummy);
%         test_imp_ids(1,cont:cont+length(incr)-1) = impostor_ids(i);
%         cont = cont+length(incr);

%some reporting
disp('Entering allID mode!')

disp('Computing client prototypes ...')
%     prototypes = zeros(size(train_ids,1),length(client_ids));
    cont=1;
    for i=1:length(client_ids);
        [incr,dummy] = find(client_ids(i) == train_ids);
        train(1:n,cont:cont+length(incr)-1) =
repmat(mean(train(1:n,dummy),2),1,length(incr));
    end

    disp('Computing similarity matrix ...')

    %compute distances
    results.match_dist = zeros(d,b);
    results.same_id = zeros(d,b);
    for i=1:d
        for j=1:b
            results.match_dist(i,j) =
return PhD_distance(train(1:n,j),test(1:n,i),dist,covar);

```

```

        if train_ids(j)==test_ids(i)
            results.same_cli_id(i,j)=1;
        else
            results.same_cli_id(i,j)=0;
        end
    end
end
results.horizontal_ids = train_ids;
results.vertical_ids = test_ids;
results.dist = dist;
results.dim = n;
disp('Done.')
end

%% This is an auxiliary function that returns the specified distance
%
% Prototype:
%   d = return_PhD_distance(x,y,dist)
%
% Inputs:
%   x       - a target feature vector
%   y       - a query feature vector
%   dist    - a string identifier determining the type similarity function
%             dist = 'cos' | 'euc' | 'ctb' | 'mahcos'
%   covar   - the inverse of the covariance matrix of the training samples
%             (required only for mahcos) - I do not perform any parameter
%             checking!!!!
%
% Outputs:
%   d       - the computed "distance" between x and y
%
% Notes:
% In each case a small distance means a similar sample a large distance
% means
% a dissimilar sample
%
function d = return_PhD_distance(x,y,dist,covar)

if nargin==3

```

```

    [a,b]=size(x);
    covar = eye(a,a);
end

%I assume that x and y are column vectors
if strcmp(dist,'euc')==1
    d = norm(x-y);
elseif strcmp(dist,'ctb')==1
    d = sum(abs(x-y));
elseif strcmp(dist,'cos')==1
    norm_x = norm(x);
    norm_y = norm(y);
    d = - (x'*y)/(norm_x*norm_y);
elseif strcmp(dist,'mahcos')==1
    norm_x = sqrt(x'*covar*x);
    norm_y = sqrt(y'*covar*y);
    d = - (x'*covar*y)/(norm_x*norm_y);
else
    disp('The specified distance is not supported!')
    return;
end

```

filter_image_with_Gabor_bank_PhD.m

```

function filtered_image =
filter_image_with_Gabor_bank_PhD(image,filter_bank,down_sampling_factor);

%% Init ops
filtered_image = [];

%% Check inputs

%check number of inputs
if nargin <2
    disp('Wrong number of input parameters! The function requires at least
two input arguments.')
    return;
elseif nargin >3
    disp('Wrong number of input parameters! The function takes at most
three input arguments.')
    return;
elseif nargin==2
    down_sampling_factor = 64;
end

%check down-sampling factor
if isnumeric(down_sampling_factor)~=1

```

```

        disp('The down-sampling factor needs to be a numeric value between
larger or equal than 1! Swithing to defaults: 64');
        down_sampling_factor=64;
    end

    if size(down_sampling_factor,1)==1 && size(down_sampling_factor,2)==1 &&
down_sampling_factor>=1
        %ok
    else
        disp('The downsampling factor needs to be a single number, greater or
equal to one!')
        return;
    end

    %check filter bank
    if isfield(filter_bank,'spatial')~=1
        disp('Could not find filters in the spatial domain. Missing
filter_bank.spatial!')
        return;
    end

    if isfield(filter_bank,'freq')~=1
        disp('Could not find filters in the frequency domain. Missing
filter_bank.freq!')
        return;
    end

    if isfield(filter_bank,'orient')~=1
        disp('Could not determine angular resolution. Missing
filter_bank.orient!')
        return;
    end

    if isfield(filter_bank,'scales')~=1
        disp('Could not determine frequency resolution. Missing
filter_bank.scales!')
        return;
    end

    %check image and filter size
    [a,b]=size(image);
    [c,d]=size(filter_bank.spatial{1,1}); %lets look at the first

    if a==2*c && d==2*b
        disp('The dimension of the input image and Gabor filters do not
match! Damn! Terminating!')
        return;
    end

    %% Compute output size
    [a,b]=size(image);
    dim_spec_down_sampl = round(sqrt(down_sampling_factor));
    new_size = [floor(a/dim_spec_down_sampl) floor(b/dim_spec_down_sampl)];

    %% Filter image in the frequency domain
    image_tmp = zeros(2*a,2*b);
    image_tmp(1:a,1:b)=image;
    image = fft2(image_tmp);

```

```

for i=1:filter_bank.scales
    for j=1:filter_bank.orient

        %filtering
        Imgabout = ifft2((filter_bank.freq{i,j}.*image));
        gabout = abs(Imgabout(a+1:2*a,b+1:2*b));

        % if you prefer to compute the real or imaginary part of the
        % filtering, uncomment the appropriate line below; the return
        % value of the function will then be changed accordingly
        %     gabout = real(Imgabout(a+1:2*a,b+1:2*b));
        %     gabout = imag(Imgabout(a+1:2*a,b+1:2*b));

        %down-sampling (the proper way to go is to use resizing
        (interpolation!!), sampling introduces high frequencies)
        y=imresize(gabout,new_size,'bilinear');

        y=(y(:)-mean(y(:)))/std(y(:)); %we use zero mean unit variance
normalization - even though histogram equalization and gaussianization
works better
        % comment out the line above and use
        % this one if you want to map a normal distribution to the filtered
        % image instead of only adjusting the mean and variance (you
        % need my INface toolbox for that)
        % y = fitt_distribution(y);
        y=y(:);

        %add to image
        filtered_image=[filtered_image;y];
    end
end

```

FaceRecognitionUsingGabor.m

```

function FaceRecognitionUsingGabor(inputFile)

    disp('Displaying input image.')
    figure(1)
    imshow(inputFile);
    disp('Press any key to continue ...')
    pause();

    trainIndex = 40;

    %% Load image database

    % construct Gabor filter bank
    filter_bank = construct_Gabor_filters_PhD(8, 5, [128 128]);

    %filter image
    proceed = 1;
    train_data = [];

```

```

ids_train = [];

try
    % construct image string, load image and extract features
    for i=1:trainIndex
        for j=1:10
            s = sprintf('database/s%i/%i.pgm',i,j);
            X = double(imread(s));
            X = imresize(X,[128 128],'bilinear');
            feature_vector =
filter_image_with_Gabor_bank_PhD(X,filter_bank,64);
            train_data = [train_data,feature_vector];
            ids_train = [ids_train;i];
        end
        disp(sprintf('Finished with feature extraction from database store
index %i', i));
    end
    [size_y,size_x] = size(X);
catch
    proceed = 0;
    disp('Could not load images from the ORL database.');
```

end

```

if(proceed)
    disp('Finished with Step 1 (database loading).')
```

%% Partitioning of the data

```

    test_data = [];
    X = double(imread(inputFile));
    X = imresize(X,[128 128],'bilinear');
    feature_vector =
filter_image_with_Gabor_bank_PhD(X,filter_bank,64);
    test_data = [test_data,feature_vector];
    disp('Finished with Step 2 (input image loading and feature
extraction).')
```

%% Construct KFA subspace

```

    model = perform_kfa_PhD(train_data, ids_train, 'fpp', [0
0.7],length(unique(ids_train))-1);
```

disp('Finished KFA subspace construction. Starting evaluation and test
image projection.')

```

    test_features = nonlinear_subspace_projection_PhD(test_data, model);
    % Compute similarity matrix
    results1 = nn_classification_PhD(model.train, ids_train, test_features,
1, size(test_features,1), 'euc','all');
```

```

    [match_score, match_ix] = min(results1.match_dist);
    disp('Found matching Images')
    display(match_ix);
    display(match_score);
```

disp('Loading similar Images of input image')

```

    i= ceil(match_ix / 10);
    for j=1:10
        s = sprintf('database/s%i/%i.pgm',i,j);
        figure(j + 1)
        imshow(s);
```

```

        end

end
disp('Finished demo.')

```

construct_Gabor_filters_PhD.m

```

function filter_bank = construct_Gabor_filters_PhD(num_of_orient,
num_of_scales, size1, fmax, ni, gamma, separation)

%% Init
filter_bank = [];

%% Check inputs

%check number of inputs
if nargin < 3
    disp('Wrong number of input parameters! The function requires at least
three input arguments.')
    return;
elseif nargin > 7
    disp('Wrong number of input parameters! The function takes no more than
seven input arguments.')
    return;
elseif nargin==3
    fmax = 0.25;
    ni = sqrt(2);
    gamma = sqrt(2);
    separation = sqrt(2);
elseif nargin==4
    ni = sqrt(2);
    gamma = sqrt(2);
    separation = sqrt(2);
elseif nargin==5
    gamma = sqrt(2);
    separation = sqrt(2);
elseif nargin==6
    separation = sqrt(2);
end

%check size
[a,b]=size(size1);
if a == 1 && b==1
    size1 = [size1 size1];
elseif a==1 && b==2
    %ok
elseif a==2 && b==1
    size1=size1'; %this is actually not needed
else
    disp('The parameter determining the size of the filters is not valid.')

```

```

        return;
    end

    %% Construct Gabor filter bank

    %init
    filter_bank.spatial = cell(num_of_scales,num_of_orient);
    filter_bank.freq = cell(num_of_scales,num_of_orient);

    %construct filters
    for u = 0:num_of_scales-1 %for each scale
        fu = fmax/(separation)^u;
        alfa = fu/gamma;
        beta = fu/ni;
        sigma_x = size1(2);
        sigma_y = size1(1);
        for v = 0:num_of_orient-1 %for each orientation
            theta_v = (v/8)*pi;
            %clear gabor
            for x=-sigma_x:sigma_x-1 %we use double the size for
frequency-domain computation
                for y=-sigma_y:sigma_y-1
                    xc = x*cos(theta_v)+y*sin(theta_v);
                    yc = -x*sin(theta_v)+y*cos(theta_v);
                    gabor(sigma_y+y+1,sigma_x+x+1)=
((fu^2)/(pi*gamma*ni))*exp(-(alfa^2*xc^2 + beta^2*yc^2))*...
                    exp((2*pi*fu*xc)*i);
                end
            end
            filter_bank.spatial{u+1,v+1} = gabor;
            filter_bank.freq{u+1,v+1}=fft2(gabor);
        end
    end

    filter_bank.scales = num_of_scales;
    filter_bank.orient = num_of_orient;

```

compute_kernel_matrix_PhD.m

```

function kermat = compute_kernel_matrix_PhD(X,Y,kernel_type,kernel_args);
%% Dummy
kermat = [];

%% Check inputs

%check number of inputs
if nargin <3
    disp('Wrong number of input parameters! The function requires at least
three input arguments.')
    return;
elseif nargin >4

```



```

        disp('Wrong number of input parameters! The function takes no more than
four input arguments.')
        return;
elseif nargin==3
    %check if the kernel type definition is valid

    %is it a string
    if ischar(kernel_type)~=1
        disp('The parameter "kernel_type" needs to be a STRING - a valid
one!')
        return;
    end

    %which one is it
    if strcmp(kernel_type,'poly')==1
        kernel_args = [0 2];
    elseif strcmp(kernel_type,'fpp')==1
        kernel_args = [0 .8];
    elseif strcmp(kernel_type,'tanh')==1
        kernel_args = 0;
    else
        disp('The entered kernel type was not recognized as a supported
kernel type.')
        return;
    end
end

%checking the validity of the kernel arguments
[a,b]=size(kernel_args);
if strcmp(kernel_type,'poly')==1
    if a==1 && b==2
        %ok
    elseif a==2 && b==1
        %ok
    else
        disp('The polynomial kernel requires the two arguments arranged
into a 1x2 matrix. Switching to default values: kernel_args = [0 2].');
        kernel_args = [0 2];
    end
elseif strcmp(kernel_type,'fpp')==1
    if a==1 && b==2
        %ok
    elseif a==2 && b==1
        %ok
    else
        disp('The fractional power polynomial kernel requires the two
arguments arranged into a 1x2 matrix. Switching to default values:
kernel_args = [0 0.8].');
        kernel_args = [0 .8];
    end
elseif strcmp(kernel_type,'tanh')==1
    if a==1 && b==1
        %ok
    else
        disp('The sigmoidal kernel requires its argument to be a single
numerical value. Switching to default: kernel_args = [0].');
        kernel_args = 0;
    end
else
    disp('The entered kernel name was not recognized as a supported kernel

```

```

type.')
    return;
end

%% Compute kernel matrices
if strcmp(kernel_type, 'poly')==1
    kermat = (X'*Y + kernel_args(1)).^(kernel_args(2));
elseif strcmp(kernel_type, 'fpp')==1
    kermat =
sign(X'*Y+kernel_args(1)).*((abs(X'*Y+kernel_args(1))).^(kernel_args(2)));
elseif strcmp(kernel_type, 'tanh')==1
    kermat = tanh(X'*Y+kernel_args(1));
end

```

DATABASE CONNECTOR: FRSDDB

C#

DbConnector.cs

```

` using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FaceRecognizerClassLibrary
{
    public class DbInteraction
    {
        static string passwordCurrent = "technicise";
        static string dbmsCurrent = "FaceRecognizerdb";

        /// <summary>
        /// Opens the database connection.
        /// </summary>
        /// <returns></returns>
        private static MySql.Data.MySqlClient.MySqlConnection OpenDbConnection()
        {
            MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;

            msqlConnection = new
            MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=" +
            passwordCurrent + ";database=" + dbmsCurrent + ";persist security info=False");

            //open the connection
            if (msqlConnection.State != System.Data.ConnectionState.Open)
                msqlConnection.Open();

            return msqlConnection;
        }
    }
}

```

```

#region User

public static int DoRegisterNewUser(UserInfo NewUser)
{
    return DoRegisterNewuserindb(NewUser);
}

private static int DoRegisterNewuserindb(UserInfo NewUser)
{
    int returnVal = 0;
    MySql.Data.MySqlClient.MySqlConnection msqlConnection =
OpenDbConnection();

    try
    {
        //define the command reference
        MySql.Data.MySqlClient.MySqlCommand msqlCommand = new
MySql.Data.MySqlClient.MySqlCommand();

        //define the connection used by the command object
        msqlCommand.Connection = msqlConnection;

        msqlCommand.CommandText = "INSERT INTO user(id,userid,passwd,hints) "
+ "VALUES(@id,@userid,@passwd,@hints)";

        msqlCommand.Parameters.AddWithValue("@id", NewUser.userId);
        msqlCommand.Parameters.AddWithValue("@userid", NewUser.userId);
        msqlCommand.Parameters.AddWithValue("@passwd", NewUser.userName);
        msqlCommand.Parameters.AddWithValue("@hints", NewUser.imageOwned);

        msqlCommand.ExecuteNonQuery();

        returnVal = 1;
    }
    catch (Exception er)
    {
        returnVal = 0;
    }
    finally
    {
        //always close the connection
        msqlConnection.Close();
    }
    return returnVal;
}

#endregion

#region ID password

public static string FetchId()
{
    string idStr = string.Empty;

    int returnVal = 0;
    MySql.Data.MySqlClient.MySqlConnection msqlConnection =
OpenDbConnection();

    try
    {

```

```

        //define the command reference
        MySql.Data.MySqlClient.MySqlCommand msqlCommand = new
MySql.Data.MySqlClient.MySqlCommand();

        //define the connection used by the command object
        msqlCommand.Connection = msqlConnection;

        msqlCommand.CommandText = "Select userid from user;";
        MySql.Data.MySqlClient.MySqlDataReader msqlReader =
msqlCommand.ExecuteReader();

        msqlReader.Read();

        idStr = msqlReader.GetString("userid");
    }
    catch (Exception er)
    {
        //Assert//.Show(er.Message);
    }
    finally
    {
        //always close the connection
        msqlConnection.Close();
    }

    return idStr;
}

public static string FetchePassword()
{
    string passwordStr = string.Empty;

    int returnVal = 0;
    MySql.Data.MySqlClient.MySqlConnection msqlConnection =
OpenDbConnection();

    try
    {

        //define the command reference
        MySql.Data.MySqlClient.MySqlCommand msqlCommand = new
MySql.Data.MySqlClient.MySqlCommand();

        //define the connection used by the command object
        msqlCommand.Connection = msqlConnection;

        msqlCommand.CommandText = "Select passwd from user;";
        MySql.Data.MySqlClient.MySqlDataReader msqlReader =
msqlCommand.ExecuteReader();

        msqlReader.Read();

        passwordStr = msqlReader.GetString("passwd");
    }
}

```

```

        catch (Exception er)
        {
            //Assert//.Show(er.Message);
        }
        finally
        {
            //always close the connection
            msqlConnection.Close();
        }

        return passwordStr;
    }
    #endregion

    #region Image

    public static int DoRegisterNewImage(StorageImageInfo ImageDetails)
    {
        int returnVal = 0;
        MySql.Data.MySqlClient.MySqlConnection msqlConnection =
        OpenDbConnection();

        try
        {
            //define the command reference
            MySql.Data.MySqlClient.MySqlCommand msqlCommand = new
            MySql.Data.MySqlClient.MySqlCommand();

            //define the connection used by the command object
            msqlCommand.Connection = msqlConnection;

            msqlCommand.CommandText = "INSERT INTO
            Image(ImageId,lastScannedDate,lastUsedHost,files,remark) "
            +
            "VALUES(@ImageId,@lastScannedDate,@lastUsedHost,@files,@filesBackupHere,@remark)";

            msqlCommand.Parameters.AddWithValue("@ImageId", ImageDetails.ImageId);
            msqlCommand.Parameters.AddWithValue("@files", ImageDetails.files);
            msqlCommand.Parameters.AddWithValue("@lastScannedDate",
            ImageDetails.lastScannedDate);
            msqlCommand.Parameters.AddWithValue("@lastUsedHost",
            ImageDetails.lastUsedHost);

            msqlCommand.ExecuteNonQuery();

            returnVal = 1;
        }
        catch (Exception er)
        {
            returnVal = 0;
        }
        finally
        {
            //always close the connection
            msqlConnection.Close();
        }
        return returnVal;
    }

    public static List<StorageImageInfo> GetAllImageList()

```

```

    {
        return QueryAllImageList();
    }
    /// <summary>
    /// Queries all Image list.
    /// </summary>
    /// <returns></returns>
    private static List<StorageImageInfo> QueryAllImageList()
    {
        List<StorageImageInfo> ImageList = new List<StorageImageInfo>();

        MySql.Data.MySqlClient.MySqlConnection msqlConnection =
OpenDbConnection();

        try
        {
            //define the command reference
            MySql.Data.MySqlClient.MySqlCommand msqlCommand = new
MySql.Data.MySqlClient.MySqlCommand();
            msqlCommand.Connection = msqlConnection;

            msqlCommand.CommandText = "Select * From Image ";
            MySql.Data.MySqlClient.MySqlDataReader msqlReader =
msqlCommand.ExecuteReader();

            while (msqlReader.Read())
            {
                StorageImageInfo Image = new StorageImageInfo();

                /*
                Image.ImageId = msqlReader.GetString("ImageId");
                Image.lastScannedDate = msqlReader.GetString("lastScannedDate");
                Image.lastUsedHost = msqlReader.GetString("lastUsedHost");
                Image.lastUsedHost = msqlReader.GetString("lastUsedHost");
                Image.files = msqlReader.GetString("files");
                */

                ImageList.Add(Image);
            }
        }
        catch (Exception er)
        {
        }
        finally
        {
            //always close the connection
            msqlConnection.Close();
        }

        return ImageList;
    }

    #endregion
}
}

```

MATLAB

OpenMySQLConnection.m

```
function conn = OpenMySQLConnection(databaseName)

%# JDBC connector path
javaaddpath('C:\Users\chandra\Documents\MATLAB\mysql-connector-java-5.0.8\mysql-connector-java-5.0.8-bin.jar')
%# connection parameteres
host = 'localhost';      %MySQL hostname
user = 'root';           %MySQL username
password = 'technicise'; %MySQL password
dbName = databaseName; %MySQL database name
%# JDBC parameters
jdbcString = sprintf('jdbc:mysql://%s/%s', host, dbName);
jdbcDriver = 'com.mysql.jdbc.Driver';

%# Create the database connection object
conn = database(dbName, user , password, jdbcDriver, jdbcString);
```

InsertIntoMySQLTable.m

```
function InsertIntoMySQLTable(conn,frontVal,topVal,sideVal)

if isconnection(conn)
    qry = sprintf('INSERT INTO faces(front,top,side)
VALUES(%s,%s,%s);',frontVal,topVal,sideVal);
    display(qry);
    fetch(exec(conn, qry));
else
    display('MySql Connection Error');
end
```

InsertIntoMySQLTableRecursive.m

```
function InsertIntoMySQLTableRecursive(conn,tableName,colArr,dataArr)

maxIndex = length(dataArr);

if isconnection(conn)
    columnFields = colArr(1);
    dataFields = dataArr(1);
    for i = 2: maxIndex
        colInput = colArr(i);
        columnFields = sprintf('%s,%s',columnFields,colInput);
        dataInput = dataArr(i);
        dataFields = sprintf('%s,%s',dataFields,dataInput);
    end
    display(columnFields);
    display(dataFields);
```

```

        qry = sprintf('INSERT INTO %s(%s)
VALUES(%s);', tableName, columnFields, dataFields);
        display(qry);
    else
        display('MySql Connection Error');
    end

```

InsertOneValIntoDB.m

```

function InsertOneValIntoDB(conn, tableName, id, description, ImgFile)

if isconnection(conn)
    qry = sprintf('INSERT INTO %s(PhotoID,Description,ImgFile)
VALUES(%s,%s, ''ImgFile'')', tableName, id, description);
    display(qry);
    fetch(exec(conn, qry));
else
    display('MySql Connection Error');
end

```

SelectAllFromDB.m

```

function alldata = SelectAllFromDB(conn, tableName)

if isconnection(conn)
    qry = sprintf('Select * From %s;', tableName);
    display(qry);
    rs = fetch(exec(conn, qry));
    alldata = get(rs, 'Data');
    display(alldata);
else
    display('MySql Connection Error');
end

```

SelectOneRowFromDB.m

```

function alldata = SelectOneRowFromDB(conn, tableName, colName, value)

if isconnection(conn)
    qry = sprintf('Select * From %s where %s =
%s;', tableName, colName, value);
    display(qry);
    rs = fetch(exec(conn, qry));
    alldata = get(rs, 'Data');
    display(alldata);
end

```



```
else
    display('MySQL Connection Error');
end
```

DATABASE CLASSES : FRSDATA

ImageInfo

```
publicclass ImageInfo
{
    publicstring imageId;
    publicstring dbpath;
    publicint size;
    publicbyte[] bytes;
}
```

UserType

```
publicenum UserType
{
    admin,
    user
}
```

UserInfo

```
publicclass UserInfo
{
    publicstring userId;
    publicstring userName;
```

```

publicStringGoogleOpenId;

publicUserType type;

List<ImageInfo> images;
}

```

FaceRecognitionState

```

publicenumFaceRecognitionState
{
FaceDetection,
FaceAlignment,
FeatureExtraction,
FeatureMatching
}

```

COMMENTS AND DESCRIPTION OF CODING SEGMENTS

CODE COMMENTING

- All comments have been written in the same language, be grammatically correct, and contain appropriate punctuation.
- Used // or /// but never /* ... */
- Did not “flowerbox” comment blocks.
- Example:

- `// *****`
- `// Comment block`
- `// *****`
- Always Used inline-comments to explain assumptions, known issues, and algorithm insights.
- Never used inline-comments to explain obvious code. Well written code is self documenting.
- Only used comments for bad code to say “fix this code” – otherwise remove, or rewrite the code!
- Included comments using Task-List keyword flags to allow comment-filtering.
- Example:
- `//always close the connection`
- `//Note.id = msqlReader.GetString("id");`
- `//define the command reference`
- Always applied C# comment-blocks (`///`) to public, protected, and internal declarations.
- Only used C# comment-blocks for documenting the API.
- Included `#region` and `#endregion` where possible for whole sections to have a `#region`-like thing and collapse them.
- Example:

```
#region User

    public static int DoRegisterNewUser(UserInfo NewUser)
    {
        return DoRegisterNewuserindb(NewUser);
    }

    private static int DoRegisterNewuserindb(UserInfo NewUser)
    {
        int returnVal = 0;

        MySql.Data.MySqlClient.MySqlConnection msqlConnection =
        OpenDbConnection();

        try
        {
            //define the command reference

            MySql.Data.MySqlClient.MySqlCommand msqlCommand = new
```

```

MySQL.Data.MySqlClient.MySqlCommand();

    //define the connection used by the command object
    msqlCommand.Connection = msqlConnection;

    msqlCommand.CommandText = "INSERT INTO
user(id,userid,passwd,hints) " + "VALUES(@id,@userid,@passwd,@hints)";

    msqlCommand.Parameters.AddWithValue("@id", NewUser.id);
    msqlCommand.Parameters.AddWithValue("@userid", NewUser.userId);
    msqlCommand.Parameters.AddWithValue("@passwd", NewUser.pass);
    msqlCommand.Parameters.AddWithValue("@hints", NewUser.hints);

    msqlCommand.ExecuteNonQuery();

    returnVal = 1;
}
catch (Exception er)
{
    returnVal = 0;
}
finally
{
    //always close the connection
    msqlConnection.Close();
}
return returnVal;
}

#endregion

```

In practical problems such as face recognition, training images of size (60x50) form a covariance matrix of size (3000x3000) that is computationally demanding in solving its eigenvalues and eigenvectors. The n-dimensional eigenvectors can be solved by first computing the eigenvectors of the (m x m, e.g. 400x400) covariance matrix, a less computationally intensive task [7]. Consider the eigenvectors v of the matrix XTX,

Premultiplying both sides by X, this results into

Where Xv_i are the eigenvectors of the original covariance matrix $C = XTX$. Therefore, it is possible to solve the eigenvectors of the original covariance (C= XTX) matrix by first getting the eigenvectors of the much smaller covariance matrix ($C = XTX$) and then multiplying it by the data matrix X, resulting to the actual eigenvectors, $v_i^* = Xv_i$.

When the number of training image samples (m, e.g. m= 400) is much less than the number of pixels (n, e.g. n= 3000) in an image, the dimension of the original feature vector ($m \ll n$), there will only be m –1, instead of n meaningful eigenvectors.

The remaining eigenvectors have an associated eigenvalues of zero. In fact, in the numerical example above, the eigenvalues of the eigenvectors, except the four chosen ones, have eigenvalues of zero.

STANDARDIZATION OF THE CODING

Coding style causes the most inconsistency and controversy between developers. Each developer has a preference, and rarely are two the same. However, consistent layout, format, and organization are key to creating maintainable code. The following sections describe the preferred way to implement C# source code in order to create readable, clear, and consistent code that is easy to understand and maintain.

FORMATTING

- Never declared more than 1 namespace per file.
- Avoided putting multiple classes in a single file.
- Always placed curly braces ({ and }) on a new line.
- Always used curly braces ({ and }) in conditional statements.
- Always used a Tab & Indention size of 4.
- Declared each variable independently – not in the same statement.
- Placed namespace “using” statements together at the top of file. Group .NET namespaces above custom namespaces.
- Grouped internal class implementation by type in the following order:

Member variables.

Constructors & Finalizers.

Nested Enums, Structs, and Classes.

Properties

Methods

- Sequence declarations within type groups based upon access modifier and visibility:

Public
Protected
Internal
Private

- Segregate interface Implementation by using #region statements.
- Append folder-name to namespace for source files within sub-folders.
- Recursively indent all code blocks contained within braces.
- Use white space (CR/LF, Tabs, etc) liberally to separate and organize code.
- Only declare related attribute declarations on a single line, otherwise stack each attribute as a separated declaration.

Example:

// Bad!

[Attribute1, Attribute2, Attribute3]

public class MyClass

{...}

// Good!

[Attribute1, RelatedAttribute2]

[Attribute3]

[Attribute4]

public class MyClass

{...}

- Place Assembly scope attribute declarations on a separate line.
- Place Type scope attribute declarations on a separate line.
- Place Method scope attribute declarations on a separate line.
- Place Member scope attribute declarations on a separate line.
- Place Parameter attribute declarations inline with the parameter.
- If in doubt, always err on the side of clarity and consistency.

CODE EFFICIENCY

We started working on the project keeping in mind that we must develop it in a way that it not only provides a very easy to use GUI but also provide a fast and flexible

service to the users. We know that a particular work can be done in more than one ways. We have tried all the options and then chose the one which provides the fastest and most secure performance. First of all, we have used the latest technologies of Microsoft like visual studio 2010 as IDE and WPF as GUI to keep our application's performance few steps ahead. We have studied all the rules of software development life cycle and applied them to keep our application flexible. We have given special attention to the storage related codes. We have avoided all the unnecessary database codes and kept them as short as possible without harming our purpose so that insertion, updating, deletion and fetching of data take place flexibly. You can see the result as a user; our application does all the works very smoothly.

ERROR HANDLING

The C# language's exception handling features help us to deal with any unexpected or exceptional situations that occur when a program is running. Exception handling uses the try, catch, and finally keywords to try actions that may not succeed, to handle failures when you decide that it is reasonable to do so, and to clean up resources afterward. Exceptions can be generated by the common language runtime (CLR), by the .NET Framework or any third-party libraries, or by application code. Exceptions are created by using the throw keyword.

In many cases, an exception may be thrown not by a method that your code has called directly, but by another method further down in the call stack. When this happens, the CLR will unwind the stack, looking for a method with a catch block for the specific exception type, and it will execute the first such catch block that it finds. If it finds no appropriate catch block anywhere in the call stack, it will terminate the process and display a message to the user.

EXCEPTIONS OVERVIEW

Exceptions have the following properties:

- Exceptions are types that all ultimately derive from `System.Exception`.

- Use a try block around the statements that might throw exceptions.

- Once an exception occurs in the try block, the flow of control jumps to the first associated exception handler that is present anywhere in the call stack. In C#, the `catch` keyword is used to define an exception handler.

- If no exception handler for a given exception is present, the program stops executing with an error message.

- Do not catch an exception unless you can handle it and leave the application in a known state. If you catch `System.Exception`, rethrow it using the throw keyword at the end of the catch block.

- If a catch block defines an exception variable, you can use it to obtain more information about the type of exception that occurred.

Exceptions can be explicitly generated by a program by using the throw keyword.

Exception objects contain detailed information about the error, such as the state of the call stack and a text description of the error.

Code in a finally block is executed even if an exception is thrown. Use a finally block to release resources, for example to close any streams or files that were opened in the try block.

Managed exceptions in the .NET Framework are implemented on top of the Win32 structured exception handling mechanism.

VALIDATION CHECKS

We have performed following data validation checks on available data:

ALLOWED CHARACTER CHECKS

Checks that ascertain that only expected characters are present in a field. For example a numeric field may only allow the digits 0-9, the decimal point and perhaps a minus sign or commas. A text field such as a personal name might disallow characters such as < and >, as they could be evidence of a markup-based security attack. An e-mail address might require exactly one @ sign and various other structural details. Regular expressions are effective ways of implementing such checks. (See also data type checks below)

BATCH TOTALS

Checks for missing records. Numerical fields may be added together for all records in a batch. The batch total is entered and the computer checks that the total is correct, e.g., add the 'Total Cost' field of a number of transactions together.

CARDINALITY CHECK

Checks that record has a valid number of related records. For example if Contact record classified as a Customer it must have at least one associated Order (Cardinality > 0). If order does not exist for a "customer" record then it must be either changed to "seed" or the order must be created. This type of rule can be complicated by additional conditions. For example if contact record in Payroll database is marked as "former employee", then this record must not have any associated salary payments after the date on which employee left organization (Cardinality = 0).

CHECK DIGITS

Used for numerical data. An extra digit is added to a number which is calculated from the digits. The computer checks this calculation when data are entered. For example the last digit of an ISBN for a book is a check digit calculated modulus 10.

CONSISTENCY CHECKS

Checks fields to ensure data in these fields corresponds, e.g., If Title = "Mr.", then Gender = "M".

CONTROL TOTALS

This is a total done on one or more numeric fields which appears in every record. This is a meaningful total, e.g., add the total payment for a number of Customers.

CROSS-SYSTEM CONSISTENCY CHECKS

Compares data in different systems to ensure it is consistent, e.g., The address for the customer with the same id is the same in both systems. The data may be represented differently in different systems and may need to be transformed to a common format to be compared, e.g., one system may store customer name in a single Name field as 'Doe, John Q', while another in three different fields: First_Name (John), Last_Name (Doe) and Middle_Name (Quality); to compare the two, the validation engine would have to transform data from the second system to match the data from the first, for example, using SQL: `Last_Name || ', ' || First_Name || substr(Middle_Name, 1, 1)` would convert the data from the second system to look like the data from the first 'Doe, John Q'

DATA TYPE CHECKS

Checks the data type of the input and give an error message if the input data does not match with the chosen data type, e.g., In an input box accepting numeric data, if the letter 'O' was typed instead of the number zero, an error message would appear.

FILE EXISTENCE CHECK

Checks that a file with a specified name exists. This check is essential for programs that use file handling.

FORMAT OR PICTURE CHECK

Checks that the data is in a specified format (template), e.g., dates have to be in the format DD/MM/YYYY.

Regular expressions should be considered for this type of validation.

HASH TOTALS

This is just a batch total done on one or more numeric fields which appears in every record. This is a meaningless total, e.g., add the Telephone Numbers together for a number of Customers.

LIMIT CHECK

Unlike range checks, data are checked for one limit only, upper OR lower, e.g., data should not be greater than 2 (≤ 2).

LOGIC CHECK

Checks that an input does not yield a logical error, e.g., an input value should not be 0 when there will be a number that divides it somewhere in a program.

PRESENCE CHECK

Checks that important data are actually present and have not been missed out, e.g., customers may be required to have their telephone numbers listed.

RANGE CHECK

Checks that the data lie within a specified range of values, e.g., the month of a person's date of birth should lie between 1 and 12.

REFERENTIAL INTEGRITY

In modern Relational database values in two tables can be linked through foreign key and primary key. If values in the primary key field are not constrained by database internal mechanism,[4] then they should be validated. Validation of the foreign key field checks that referencing table must always refer to a valid row in the referenced table.

SPELLING AND GRAMMAR CHECK

Looks for spelling and grammatical errors.

UNIQUENESS CHECK

Checks that each value is unique. This can be applied to several fields (i.e. Address, First Name, Last Name).

TABLE LOOK UP CHECK

A table look up check takes the entered data item and compares it to a valid list of entries that are stored in a database table.

TESTING

TESTING TECHNIQUES AND TESTING STRATEGIES USED

FRS application will be tested using following strategies to ensure that the application succeeds to complete all the functional and non functional requirements:

DATABASE & DATA INTEGRITY TESTING

The databases and the database processes should be tested as a subsystem within the **FRS** Application. These subsystems should be tested with the target-of-test's User Interface as the interface to the database.

Test Objective:	Ensure that data is stored correctly, audits can be performed, access is controlled
Technique:	SQL queries will be executed in the DB to verify the data content and correctness.
Completion Criteria:	All planned tests have been executed. All defects that have been identified have been resolved All resolutions have been implemented.

FUNCTIONAL TESTING:

Function testing focuses on any requirements for test that can be traced directly to use cases or business functions and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques; that are verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analyzing the output or results. Identified below is an outline of the function testing recommended for **FRS**:

Test Objective:	Ensure proper target-of-test functionality, including business process validation.
-----------------	--

Technique:	<p>Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following:</p> <ul style="list-style-type: none"> • The expected results occur when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used. • Business rules are properly applied. • Black Box end to end testing of configured processes. Manual validation of required and optional fields.
Completion Criteria:	<ul style="list-style-type: none"> • All planned tests have been executed. • All defects that have been identified have been resolved • All resolutions have been implemented.

REGRESSION TESTING:

Regression testing focuses on software functionality that may have been previously working however through subsequent changes may have been inadvertently impacted. The goals of these tests are to verify that the broader impact of changes has been verified. Identified below is an outline of the regression testing recommended for each application(s)/module(s) of **FRS**.

Test Objective:	Ensure that previously passed test cases continue to pass as the new system development is deployed and that surrounding systems that may be impacted by a change are still functioning as expected.
Technique:	<ul style="list-style-type: none"> • Execute previous passed testing suites to ensure the following: • The expected results occur when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used. • Each business rule is properly applied.
Completion Criteria:	<ul style="list-style-type: none"> • All planned regression tests have been executed. • All identified defects have been resolved.

USER INTERFACE TESTING:

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards. Most of this testing will have been done during functional testing. The areas of focus will be on design, layout and navigation of the screens.

Test Objective:	UI testing will verify the screens and the layouts and navigation
Technique:	<ul style="list-style-type: none"> • Verify the design and layout of the screen. • Identify the integration links. • Test the functioning of the links – that the proper page is displayed and correct messages, pop-ups are shown when they need to be displayed etc • Validation of general navigation
Completion Criteria:	<ul style="list-style-type: none"> • All navigation test cases have been executed. • All screens have been verified as per design and layouts • All defects that have been identified have been resolved.

PERFORMANCE PROFILING:

Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune performance behaviours as a function of conditions such as workload or hardware configurations

Test Objective:	The purpose of performance profiling is to ensure the performance of the FRS application is up to the desired level.
Technique:	<ul style="list-style-type: none"> • Use a subset of Test Procedures developed for Function and Business Cycle Testing. • Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs. • This will be done by using Load Runner or Quick Test Professional (QTP).

Completion Criteria:	<ul style="list-style-type: none"> • Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction. • Results are recorded and a performance baseline is created for the major logical functions within the scenarios listed above. • All performance defects are reviewed and triaged to an acceptable resolution.
----------------------	--

LOAD TESTING:

Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviours and ability of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly at the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time sensitive issues.

Test Objective:	The purpose of load testing is to verify performance behaviour time for designated transactions or business cases under varying workload conditions.
Technique:	<ul style="list-style-type: none"> • Use a subset of Test Procedures developed for Function and Business Cycle Testing. • Scripts will be executed to simulate the peak load for 1 hour and report will be generated and analysed. • This will be done using Load Runner.
Completion Criteria:	<ul style="list-style-type: none"> • Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation. • Results are recorded and a performance baseline is created for the major business functions within the scenarios listed above. • All load testing defects are reviewed and triaged to an acceptable resolution.

STRESS TESTING:

Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions. Other defects might result from competition for shared resources like database locks or network bandwidth. Stress testing can also be used to

identify the peak workload the target-of-test can handle, which is often beyond the production workload.

VOLUME TESTING:

Volume Testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume Testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.

SECURITY & ACCESS CONTROL TESTING:

Security and Access Control Testing focus on following key areas of security:

- Application-level security, including access to the Data or Business Functions

Application-level security ensures the authentication and authorization of a user. Authentication ensures that the user is a valid user of the system and authorization ensures that the user has the proper privileges to perform specific tasks on desired resources of the system. Testing will be conducted to validate the rules by taking into considerations the various roles applicable for the system.

FAILOVER & RECOVERY TESTING:

Failover and Recovery Testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity.

Failover testing ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without loss of data or transactions.

Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/Output (I/O) failures or invalid database pointers and keys. Recovery processes are invoked and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.

CONFIGURATION TESTING:

Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client

workstations, network connections and database servers vary. Client workstations may have different software loaded—for example, applications, drivers, and so on—and at any one time, many different combinations may be active using different resources.

INSTALLATION/DEPLOY & BACK OUT TESTING:

Installation testing has two purposes. The first is to ensure that the software can be installed under different conditions—such as a new installation, an upgrade and a complete or custom installation—under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, and so on. The second purpose is to verify that, once installed; the software operates correctly and can be backed out successfully. This usually means running a number of the tests that were developed for Function testing before and after the back out.

POST PRODUCTION TESTING:

The purpose of Post production testing is to verify that, once deployed, the software operates correctly. This usually means running a number of the tests that were developed for Function Testing ensuring that no data is changed/modified in production. Typically, the business SME's assist with Post production testing.

UNIT TESTING:

Unit testing will take place within the construction phase of the project. After application module has been built to meet design specifications, each component (screen, view, interface, conversion program, etc.) will be tested individually to help confirm that it functions properly as an individual unit. Basic performance testing will also be completed during unit test to resolve obvious issues with performance prior to the System Testing.

The resource responsible for development will conduct testing of their module using the unit test conditions defined by the developer based on detailed design documents. The final step of unit test will be a review by the team lead to obtain their signoff on the component test checklist.

SMOKE TESTING:

Test Objective:	Verifies the major functionality at high level in order to determine if further testing is possible.
Technique:	<ul style="list-style-type: none">• After initial deployment to the test environment validate all critical components of the application prior to proceeding with testing.

Completion Criteria:	<ul style="list-style-type: none"> Navigation through the application at high level is possible, testing can continue.
----------------------	---

DATA MIGRATION TESTING:

This is the process of testing to verify whether or not the data migration (or conversion) has been successfully completed. The testing process will be carried out by running SQL scripts on both the source and destination databases.

The fields which are present in the newdata Model in the Destination DB(s) will be migrated from the existing systemssource DB(s) to Destination DB(s).

Test Objective:	The objective of this test is to verify that data migration is successful from source DB(s) to destination DB(s).
Technique:	<ul style="list-style-type: none"> The Team is notified before the data migration. Team runs queries on the source DB and fetches the data. Data Migration is done. Mapped data needs to be determined. Team runs the queries on the Destination DB and fetches the data. Cross verification of the data is done to see that data fetched from the old database is same as the data fetched from the new database. Verification of the table structure. Verification of record counts. Verification of the data formatting.
Completion Criteria:	<ul style="list-style-type: none"> Data fetched from the Source DB(s) and Destination DB(s) matches. The record count in the Source and the Destination databases should be equal. No data are truncated. Data formatting is proper (if required at any instance). All defects that have been identified have been resolved.

TESTING PLAN USED

Testing is a set of activities that can be planned in advance and conducted systematically. During testing, the program to be tested is executed against a set of test cases, and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to.

In a software development project, errors can be injected at any stage during the development. Some requirement errors and design errors are likely to remain undetected. Ultimately, these errors will be reflected in the code. Hence, testing performs a very critical role for quality assurance and for ensuring the reliability of software.

CAUSE OF TESTING:

The first test of the system is to see whether it produces the correct output. Following this a variety of other tests are conducted:

Response time: this test is conducted to measure the processing of the software.

Volume testing: In this test, we create as much data as would normally be used to a variety that the hardware and the software will function correctly.

Stress Testing: The purpose of this is to prove that the candidate system should not malfunction under peak load.

Recovery & Security: A forced system is induced to test a backup recovery procedure.

System testing is verification of the workability of a system. For this purpose the system is used experimentally to ensure that it will run according to its specification and in the way users expect. These are two stages to this:

- ❖ The testing of the individual programs by their programmers called Unit testing and
- ❖ The testing of the overall System testing.

UNIT TESTING INCLUDES THE FOLLOWING:

- ❖ Test for number of input parameters equal to number of arguments.
- ❖ Test the parameter and argument attributes match.
- ❖ Feasibility and validity checks on input data.
- ❖ Checks for interpretation of symbols correctly.
- ❖ Checks for accurate branching and looping.
- ❖ Checks on logical file addressing and searching.
- ❖ Checks on the capacity of storage areas and buffers.
- ❖ Checks on updating procedure.
- ❖ Checks on contents and layout of printed and displayed output.
- ❖ Checks for batch control totals.
- ❖ Checks on interfacing with other programs, software, database and operating system.
- ❖ Checks on documentation.

SYSTEM TESTING INCLUDES THE FOLLOWING:

- ❖ Interfacing of run within the system.
- ❖ Compilation and continuity of control totals.
- ❖ Capacity of logical file storage areas and the handling of overflow.
- ❖ Error correction procedures including user involvement.
- ❖ User request for amendments and output.

- ❖ Timing of runs and routines for the data volumes to be actually handled.
- ❖ Output preparation and distribution.
- ❖ Audit requirements.
- ❖ Logical physical file housekeeping and control.

The usual procedures in testing are to create data for the initial tests and to use live data for later testing.

THE FOLLOWING POINTS SHOULD BE REMEMBERED PRIMARILY:

- ❖ Both the artificial and live data should be representative of reality;
- ❖ Live data can often be checked against the previous system's result;
- ❖ If the previous and new system differ, the two sets of result should be reconciled if at all possible;
- ❖ Logical files are usually needed to fully test the programs and routines;
- ❖ Data generating techniques are useful for simulating large volume of input data file records;
- ❖ In the final trial run of the complete routine, asset of input data is passed through to the resultant output and/or file updating stage(s);
- ❖ Test data should include known incorrect data with a view to test the validation and control procedure.

ANY ENGINEERED PRODUCT (AND MOST OTHER THINGS) CAN BE TESTED IN ONE OF TWO WAYS:

- ❖ Knowing the specified function that a product has been designed to perform, test can be conducted that demonstrate each function is fully operational, at the same time searching for errors in each function;
- ❖ Knowing the internal workings of a product, tests can be conducted to ensure that all internal operation performs according to specification and all internal components have been adequately exercised. The first test approach is called Black-box and the second, White-box testing.

WHITE-BOX TESTING:

White-box testing sometimes called glass-box testing, is a test case design to derive test cases. Using white-box testing method, the software engineer can test:

- ❖ Guarantee that all independent paths within a module have been exercised at least one;
- ❖ Exercise all logical decisions on their true and false sides;
- ❖ Execute all loops at there boundaries and within there operational bounds; and
- ❖ Exercise internal data structures to assure there validity.

A reasonable question might be posed at this juncture: “why spend time and energy worrying about (and testing) logical minutes when we might better spend effort ensuring that program requirements have been met?” stated another way, why don’t we spend all of our energies on black-box testing? The answer lies in the nature of software defects.

- Logical errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed. Errors tend to creep into our work when we design and implement function, conditions and control that are out of the mainstream. Every processing tends to be well understood (and well scrutinized), while “special case” processing tends to fall into the cracks.
- We often believe that a logical path is not likely to be executed when, in fact, it may be executed in regular basis. The logical flow of a program is sometimes counterintuitive, meaning that our unconscious assumption about flow of control and

data may lead us to make design errors that are uncovered only when path testing commences.

- Typographical errors are random. When a program is translated into programming language source code, it is likely that some typing error will occur. Many will be uncovered by syntax checking mechanism, but others will go undetected until testing begins. It is likely that a type will exist on an obscure logical path as on a mainstream path.

Each of these reasons provides an argument for conducting white-box tests. Black-box testing, no matter how thorough, may miss the kinds of errors noted above. As Beizer has stated: “Bugs lurk in corners and congregate at boundaries”. White-box testing is far more likely to uncover them.

BASIS PATH TESTING:

Basis path testing is a **White-box testing** technique first proposed by Tom McCabe. The basis path methods enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basic set that are guaranteed to execute every statement in the program at least one time during testing.

BLACK-BOX TESTING:

It focuses on the functional requirements of the software. That is Black-box testing enables software engineer to derive sets of input conditions that will fully exercise all functional requirements of a program. Black-box testing is not a alternative of white box testing. Rather, it is a complementary approach that is likely to uncover a different class of errors than White-box methods.

Black-box testing attempts to find errors in the following categories.

- ❖ Incorrect or missing function.
- ❖ Interface errors
- ❖ Errors in data structures or external data base access
- ❖ Performance errors
- ❖ Initializations or termination errors

Unlike **White-box** testing, which is performed early in the testing process, **black-box** testing tends to be applied during later stage of testing. Because **black-box** testing purposely disregards control structure, attention is focused on the information domain. Tests are based on source data from a

previous period so that the result from the new system can be compared with that of the old one.

With those of the previous ones to answer the following questions:

- ❖ How is functional validity testing?
- ❖ What classes of input will make good test cases?
- ❖ Is the system particularly sensitive to certain input values?
- ❖ How are the boundaries of a data class isolated?
- ❖ What data rates and data volume can the system tolerate?
- ❖ What effect will specific combinations of the data have on system operation?

By applying **Black-box** techniques, we derive a set of test cases that satisfy the following criteria:

- ❖ Test cases that reduce, by errors and a designed to achieve reasonableness testing.
- ❖ Test cases that tell us something about the presence or absence of classes of errors, than errors associated only with the specific test at hand.

TEST REPORTS FOR UNIT TEST CASES AND SYSTEM TEST CASES

TEST REPORTS FOR UNIT TEST CASES

Test Case Id	Comment	Status
FRS-001	NA	PASS
FRS-002	NA	PASS
FRS-003	NA	PASS
FRS-004	NA	PASS
FRS-005	NA	PASS
FRS-006	NA	PASS
FRS-007	NA	PASS
FRS-008	NA	PASS
FRS-009	NA	PASS
FRS-010	NA	PASS
FRS-011	NA	PASS
FRS-012	NA	PASS
FRS-013	NA	PASS
FRS-014	NA	PASS
FRS-015	NA	PASS
FRS-016	NA	PASS
FRS-017	NA	PASS

FRS-018	NA	PASS
FRS-019	NA	PASS
FRS-020	NA	PASS
FRS-021	NA	PASS
FRS-022	NA	PASS
FRS-023	NA	PASS
FRS-024	NA	PASS
FRS-025	NA	PASS
FRS-026	NA	PASS
FRS-027	NA	PASS
FRS-028	NA	PASS
FRS-029	NA	PASS
FRS-030	NA	PASS
FRS-031	NA	PASS
FRS-032	NA	PASS
FRS-033	NA	PASS
FRS-034	NA	PASS
FRS-035	NA	PASS
FRS-036	NA	PASS
FRS-037	NA	PASS
FRS-038	NA	PASS
FRS-039	NA	PASS
FRS-040	NA	PASS
FRS-041	NA	PASS
FRS-042	NA	PASS
FRS-043	NA	PASS
FRS-044	NA	PASS
FRS-045	NA	PASS

FRS-046	NA	PASS
FRS-047	NA	PASS
FRS-048	NA	PASS
FRS-049	NA	PASS
FRS-050	NA	PASS
FRS-051	NA	PASS
FRS-052	NA	PASS
FRS-053	NA	PASS
FRS-054	NA	PASS
FRS-055	NA	PASS
FRS-056	NA	PASS
FRS-057	NA	PASS
FRS-058	NA	PASS
FRS-059	NA	PASS
FRS-060	NA	PASS
FRS-061	NA	PASS
FRS-062	NA	PASS
FRS-063	NA	PASS
FRS-064	NA	PASS
FRS-065	NA	PASS
FRS-066	NA	PASS

TEST REPORTS FOR SYSTEM TEST CASES

Test Case Id	Comment	Status
FRS-067	NA	PASS

FRS-068	NA	PASS
FRS-069	NA	PASS
FRS-070	NA	PASS
FRS-071	NA	PASS
FRS-072	NA	PASS
FRS-073	NA	PASS
FRS-074	NA	PASS
FRS-075	NA	PASS
FRS-076	NA	PASS
FRS-077	NA	PASS
FRS-078	NA	PASS
FRS-079	NA	PASS
FRS-080	NA	PASS
FRS-081	NA	PASS
FRS-082	NA	PASS
FRS-083	NA	PASS
FRS-084	NA	PASS
FRS-085	NA	PASS
FRS-086	NA	PASS
FRS-087	NA	PASS
FRS-089	NA	PASS
FRS-090	NA	PASS
FRS-091	NA	PASS
FRS-092	NA	PASS
FRS-093	NA	PASS

The steps in the bellow section demonstrate how to create a console application that uses the `Debug` class to provide information about the program execution.

When the program is run, we can use methods of the `Debug` class to produce messages that help we to monitor the program execution sequence, to detect malfunctions, or to provide performance measurement information. By default, the messages that the `Debug` class produces appear in the Output window of the Visual Studio Integrated Development Environment (IDE).

The sample code uses the `WriteLine` method to produce a message that is followed by a line terminator. When we use this method to produce a message, each message appears on a separate line in the Output window.

When we use the `Assert` method of the `Debug` class, the Output window displays a message only if a specified condition evaluates to false. The message also appears in a modal dialog box to the user. The dialog box includes the message, the project name, and the `Debug.Assert` statement number. The dialog box also includes the following three command buttons:

- **Abort:** The application stops running.
- **Retry:** The application enters debug mode.
- **Ignore:** The application proceeds.

The user must click one of these buttons before the application can continue.

We can also direct output from the **Debug** class to destinations other than the Output window. The **Debug** class has a collection named **Listeners** that includes **Listener** objects.

Each **Listener** object monitors **Debug** output and directs the output to a specified target.

Each **Listener** in the **Listener** collection receives any output that the **Debug** class generates. Use the **TextWriterTraceListener** class to define **Listener** objects. We can specify the target for a **TextWriterTraceListener** class through its constructor.

Some possible output targets include the following:

- The Console window by using the **System.Console.Out** property.
- A text (.txt) file by using the **System.IO.File.CreateText("FileName.txt")** statement.

After we create a **TextWriterTraceListener** object, we must add the object to the **Debug.Listeners** collection to receive Debug output.

CREATE A SAMPLE WITH THE DEBUG CLASS

1. Start Visual Studio or Visual C# Express Edition.
2. Create a new Visual C# Console Application project named **conInfo**. Class1 is created in Visual Studio .NET. Program.cs is created in Visual Studio 2005.
3. Add the following namespace at top in Class1 or Program.cs.

```
using System.Diagnostics;
```

4. To initialize variables to contain information about a product, add the following declaration statements to **Main** method:

```
5. string sProdName = "Widget";
```

```
6. int iUnitQty = 100;
```

```
double dUnitCost = 1.03;
```

7. Specify the message that the class produces as the first input parameter of the **WriteLine** method. Press the CTRL+ALT+O key combination to make sure that the Output window is visible.

```
Debug.WriteLine("Debug Information-Product Starting ");
```

8. For readability, use the **Indent** method to indent subsequent messages in the Output window:

```
Debug.Indent();
```

9. To display the content of selected variables, use the **WriteLine** method as follows:

```
10. Debug.WriteLine("The product name is " + sProdName);
```

```
11. Debug.WriteLine("The available units on hand are " + iUnitQty.ToString());
```

```
Debug.WriteLine("The per unit cost is " + dUnitCost.ToString());
```

12. We can also use the **WriteLine** method to display the namespace and the class name for an existent object. For example, the following code displays the **System.Xml.XmlDocument** namespace in the Output window:

```
13. System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();
```

```
Debug.WriteLine(oxml);
```

14. To organize the output, we can include a category as an optional, second input parameter of the **WriteLine** method. If we specify a category, the format of the Output window message is "category: message." For example, the first line of the following code displays "Field: The product name is Widget" in the Output window:

15. `Debug.WriteLine("The product name is " + sProdName,"Field");`
16. `Debug.WriteLine("The units on hand are" + iUnitQty,"Field");`
17. `Debug.WriteLine("The per unit cost is" + dUnitCost.ToString(),"Field");`
`Debug.WriteLine("Total Cost is " + (iUnitQty * dUnitCost),"Calc");`
18. The Output window can display messages only if a designated condition evaluates to true by using the **WriteLineIf** method of the **Debug** class. The condition to be evaluated is the first input parameter of the **WriteLineIf** method. The second parameter of **WriteLineIf** is the message that appears only if the condition in the first parameter evaluates to true.
19. `Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");`
20. `Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");`
21. Use the **Assert** method of the **Debug** class so that the Output window displays the message only if a specified condition evaluates to false:
22. `Debug.Assert(dUnitCost > 1, "Message will NOT appear");`
23. `Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost < 1 is false");`
24. Create the **TextWriterTraceListener** objects for the Console window (tr1) and for a text file named Output.txt (tr2), and then add each object to the **Debug Listeners** collection:
25. `TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);`
26. `Debug.Listeners.Add(tr1);`
- 27.
28. `TextWriterTraceListener tr2 = new
 TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));`
`Debug.Listeners.Add(tr2);`
29. For readability, use the **Unindent** method to remove the indentation for subsequent messages that the **Debug** class generates. When we use the **Indent** and the **Unindent** methods together, the reader can distinguish the output as group.
30. `Debug.Unindent();`
`Debug.WriteLine("Debug Information-Product Ending");`
31. To make sure that each **Listener** object receives all its output, call the **Flush** method for the **Debug** class buffers:
`Debug.Flush();`

USING THE TRACE CLASS

We can also use the **Trace** class to produce messages that monitor the execution of an application. The **Trace** and **Debug** classes share most of the same methods to produce output, including the following:

- **WriteLine**
- **WriteLineIf**
- **Indent**
- **Unindent**
- **Assert**
- **Flush**

We can use the **Trace** and the **Debug** classes separately or together in the same application. In a Debug Solution Configuration project, both **Trace** and **Debug** output are active. The project generates output from both of these classes to all **Listener** objects. However, a Release Solution Configuration project only generates output from a **Trace** class. The Release Solution Configuration project ignores any **Debug** class method invocations.

```
Trace.WriteLine("Trace Information-Product Starting ");
Trace.Indent();

Trace.WriteLine("The product name is "+sProdName);
Trace.WriteLine("The product name is "+sProdName,"Field" );
Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Trace.Assert(dUnitCost > 1, "Message will NOT appear");

Trace.Unindent();
Trace.WriteLine("Trace Information-Product Ending");

Trace.Flush();

Console.ReadLine();
```

VERIFY THAT IT WORKS

1. Make sure that **Debug** is the current solution configuration.

2. If the **Solution Explorer** window is not visible, press the CTRL+ALT+L key combination to display this window.
3. Right-click **conInfo**, and then click **Properties**.
4. In the left pane of the conInfo property page, under the **Configuration** folder, make sure that the arrow points to **Debugging**.

Note In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Debug** in the **conInfo** page.

5. Above the **Configuration** folder, in the **Configuration** drop-down list box, click **Active (Debug)** or **Debug**, and then click **OK**. In Visual C# 2005 and in Visual C# 2005 Express Edition, click **Active (Debug)** or **Debug** in the **Configuration** drop-down list box in the **Debug** page, and then click **Save** on the **File** menu.
6. Press CTRL+ALT+O to display the Output window.
7. Press the F5 key to run the code. When the **Assertion Failed** dialog box appears, click **Ignore**.
8. In the Console window, press ENTER. The program should finish, and the Output window should display the output that resembles the following

9. Debug Information-Product Starting
10. The product name is Widget
11. The available units on hand are100
12. The per unit cost is 1.03
13. System.Xml.XmlDocument
14. Field: The product name is Widget
15. Field: The units on hand are100
16. Field: The per unit cost is1.03
17. Calc: Total Cost is 103
18. This message WILL appear
19. ---- DEBUG ASSERTION FAILED ----
20. ---- Assert Short Message ----
21. Message will appear since dUnitcost < 1 is false
22. ---- Assert Long Message ----
- 23.
- 24.


```
25.  at Class1.Main(String[] args) <%Path%>\class1.cs(34)
26.
27.  The product name is Widget
28.  The available units on hand are100
29.  The per unit cost is 1.03
30. Debug Information-Product Ending
31. Trace Information-Product Starting
32.  The product name is Widget
33.  Field: The product name isWidget
34.  This message WILL appear
35. Trace Information-Product Ending
36.
```

37. The Console window and the Output.txt file should display the following output:

```
38. The product name is Widget
39.  The available units on hand are 100
40.  The per unit cost is 1.03
41. Debug Information-Product Ending
42. Trace Information-Product Starting
43.  The product name is Widget
44.  Field: The product name is Widget
45.  This message WILL appear
46. Trace Information-Product Ending
```

Note The Output.txt file is located in the same directory as the conInfo executable (conInfo.exe). Typically, this is the \bin folder where the project source is stored. By default, this is C:\Documents and Settings*User login*\My Documents\Visual Studio Projects\conInfo\bin. In Visual C# 2005 and in Visual C# 2005 Express Edition, the Output.txt file is located in the following folder:

C:\Documents and Settings*User login*\My Documents\Visual Studio
2005\Projects\conInfo\conInfo\bin\Debug

COMPLETE CODE LISTING

```
using System;

using System.Diagnostics;

class Class1
{
    [STAThread]
    static void Main(string[] args)
    {
        string sProdName = "Widget";
        int iUnitQty = 100;
        double dUnitCost = 1.03;

        Debug.WriteLine("Debug Information-Product Starting ");
        Debug.Indent();

        Debug.WriteLine("The product name is "+sProdName);
        Debug.WriteLine("The available units on hand are"+iUnitQty.ToString());
        Debug.WriteLine("The per unit cost is "+ dUnitCost.ToString());

        System.Xml.XmlDocument oxml = new System.Xml.XmlDocument();
        Debug.WriteLine(oxml);

        Debug.WriteLine("The product name is "+sProdName,"Field");
        Debug.WriteLine("The units on hand are"+iUnitQty,"Field");
        Debug.WriteLine("The per unit cost is "+dUnitCost.ToString(),"Field");
        Debug.WriteLine("Total Cost is "+(iUnitQty * dUnitCost),"Calc");
```

```

Debug.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Debug.WriteLineIf(iUnitQty < 50, "This message will NOT appear");

Debug.Assert(dUnitCost > 1, "Message will NOT appear");
Debug.Assert(dUnitCost < 1, "Message will appear since dUnitcost < 1 is false");

    TextWriterTraceListener tr1 = new TextWriterTraceListener(System.Console.Out);
Debug.Listeners.Add(tr1);

    TextWriterTraceListener tr2 = new
TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));
Debug.Listeners.Add(tr2);

Debug.WriteLine("The product name is "+sProdName);
Debug.WriteLine("The available units on hand are"+iUnitQty);
Debug.WriteLine("The per unit cost is "+dUnitCost);
Debug.Unindent();
Debug.WriteLine("Debug Information-Product Ending");
Debug.Flush();

Trace.WriteLine("Trace Information-Product Starting ");
Trace.Indent();

Trace.WriteLine("The product name is "+sProdName);
Trace.WriteLine("The product name is "+sProdName,"Field" );
Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Trace.Assert(dUnitCost > 1, "Message will NOT appear");

```

```

Trace.Unindent();

Trace.WriteLine("Trace Information-Product Ending");


Trace.Flush();


Console.ReadLine();

    }

}

```

TROUBLESHOOT

- If the solution configuration type is **Release**, the **Debug** class output is ignored.
- After we create a **TextWriterTraceListener** class for a particular target, **TextWriterTraceListener** receives output from the **Trace** and the **Debug** classes. This occurs regardless of whether we use the **Add** method of the **Trace** or the **Debug** class to add **TextWriterTraceListener** to the **Listeners** class.
- If we add a **Listeners** object for the same target in the **Trace** and the **Debug** classes, each line of output is duplicated, regardless of whether **Debug** or **Trace** generates the output.

```

    TextWriterTraceListener myWriter = new
    TextWriterTraceListener(System.Console.Out);

    Debug.Listeners.Add(myWriter);


    TextWriterTraceListener myCreator = new
    TextWriterTraceListener(System.Console.Out);

    Trace.Listeners.Add(myCreator);

```

SYSTEM SECURITY MEASURES:

DATABASE/DATA SECURITY:

It encrypts the data stored in the database so that even if someone succeeds to hack the database still not much harm could be done.

The application will use Google open-id authentication for web interface.

CREATION OF USER PROFILES AND ACCESS RIGHTS

The software requires a predefined username and password to login.

It allows a guest login as well which lets a guest user this application with very limited access to the user data.

COST ESTIMATION OF THE PROJECT ALONG WITH COST ESTIMATION MODEL

Software development is a highly labor intensive activity. A project of large dimension can easily turn into chaos if proper management controls are not imposed. Therefore the cost/expenditure and the profit gained after implementing the project has to be taken into account. That is we have to consider the cost benefit analysis.

This cost/benefit may be tangible or intangible, direct or indirect, fixed or variable. To build up a large software all the elements required, are estimated to get the development cost of the considering project. When we consider all this requirements we can develop a cost estimation model to find proposed cost of the developing project. And from this model we can track down the expenditure during the course of development.

Now after implementing the project we have to consider gain from it in terms of benefits, that is how much person month can be saved from this project. Therefore we have to consider the total expenditure and the benefit gain from the project once it has been implemented. Here we express the benefits in the terms of person month that is monthly salary of the person concerned for the system, which has to be replaced. Therefore this cost/benefit analysis report gives us a total picture of how a company gets benefit from candidate system once that has replace a older one.

The project developing components like hardware, personnel, facility and supply cost are also taken into consideration during the cost estimation. Then we identify the cost and benefit of a given system and categories them or analysis. And from that estimated cost we track the expenditure and then calculate the benefits.

In developing the cost estimation of a project we need to consider several cost elements. Among them is hardware, personal, facility, operating and supply cost are noteworthy.

The model for estimating cost is mainly based on the total lines of code delivered. As this is not such code based rather than a plug-in based project so we estimate the cost on the consideration of how much time it can take in plugging the user interfaces and how many interfaces are required. The cost is then calculated from the total plug-in hours and as it is a single handed project, so this is the time taken by a single person.

The cost of man-power involved in this project is not considered in this estimation. We should consider the cost when we shall go for any live project. This cost is depending on several factors like skill set, location of country etc. e.g. man-hour cost is around Rs.250.00 to Rs.300.00 in India whereas for USA it varies from US\$60.00 to US\$200.00. Most of the cases, Man (person) power cost is considerable higher than all other costs. Software cost and effort estimation will never be an exact science. Too many variables human, technical, environmental can affect the ultimate cost of the software and effort applied to develop it. To achieve reliable cost and effort estimates, a number of options arise, 1) Base estimates on similar projects that have been already completed; 2) Using relatively simple “decomposition techniques” to generate project cost and effort estimates; 3) Using one or more empirical models for software cost and effort estimation.

We used the basic COCOMO model, which gives an approximate estimate of our **FRS** project parameters. The basic COCOMO estimation model is given by the following expressions:

$$\text{Effort} = a_1 * (\text{KLOC})^{a_2} \text{ PM}$$

$$\text{Tdev} = b_1 * (\text{Effort})^{b_2} \text{ months}$$

Where

KLOC is the estimated size of the software product expressed in Kilo Lines of Code a_1 , a_2 , b_1 , b_2 are constants for each category of software products.

Tdev is the estimated time to develop the software, expressed in months.

Effort is the total effort required to develop the software product, expressed in person-month (PM).

Our project is semidetached type, because the development team consists of a mixture of experienced and inexperienced staff like my guide and me. Team members may have limited

experience on related system but may be unfamiliar with aspects of the system being developed.

ESTIMATION OF DEVELOPMENT EFFORT

For our Semi-detached class software product **FRS**, the formula for estimating the effort based on the code size is shown below:

Semi-detached **FRS**: $T_{dev} = 3.0 * (KLOC)^{1.12}$ PM

ESTIMATION OF DEVELOPMENT TIME

For our Semi-detached class software product **FRS**, the formula for estimating the development time based on the effort is given below:

Semi-detached **FRS**: $T_{dev} = 2.5 * (Effort)^{0.35}$ months

Assume that the size of a Semi-detached **FRS** product has been estimated to be 3,200 lines of source code. Assume that the average salary of software engineer(me) is Rs. 20,000 per month.

Assume that the size of our

The basic COCOMO estimation formula for **FRS** semidetached software:

Our Effort $= 3.0 * (3.2)^{1.12}$ PM

$= 11$ PM

Normal Development time $= 2.5 * (11)^{0.35}$ months

$= 6$ months

Cost required to develop the product $= Rs. 6 * 20000$

$= Rs. 120,000$

REPORTS

- List of Users updates could be generated.

- List of users logged on update could be generated.
- A list of usage report could be generated.
- List of failed logged in attempt update could be generated.

FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT

- Now it will display the text based RSS feeds and link of the multimedia contents. We will display the Multimedia contents like Video, Audio & Image in future.
- To support UNIX / Linux Based Operating systems.
- To Support Mobile Operating systems for Symbian, Meego& Android.

BIBLIOGRAPHY

WEBSITE

- <http://en.wikipedia.org>
- http://en.wikipedia.org/wiki/Windows_Presentation_Foundation
- <http://msdn.microsoft.com/en-us/>
- <http://www.c-sharpcorner.com/beginners/>
- <http://www.microsoft.com/en-us/default.aspx>
- <http://www.codeplex.com/>
- <http://stackoverflow.com/>
- <http://www.codeguru.com/>
- <http://www.csharpcourse.com/>
- <http://www.w3schools.com>
- <http://blogs.technicise.com/>
- <http://connect.technicise.com/>

- <http://www.mysql.com/>
- <http://www.mysql.com/support/>
- <http://dev.mysql.com/>
- <http://dev.mysql.com/support/>
- <http://www.homeandlearn.co.uk/csharp/csharp.html>
- <http://www.wpftutorial.net/Home.html>
- <https://www.facebook.com/>
- <https://developers.facebook.com/>
- <https://developers.facebook.com/apps>
- <https://www.facebook.com/creatormyapp>
- <https://www.facebook.com/help/>
- <http://nuget.codeplex.com/>
- <https://twitter.com/#>
- <https://dev.twitter.com/>
- <https://dev.twitter.com/discussions/3477>
- <https://support.twitter.com/>
- <https://developers.google.com/>
- <https://maps.google.co.in/>
- <http://support.google.com/maps/?hl=en&rd=1>
- <http://www.youtube.com/?gl=IN>
- <http://blendinsider.com/>
- <https://github.com/>
- <https://github.com/anirban-nandy>
- <https://github.com/anirban-nandy/FaceRecognition>
- <http://learn.github.com/p/intro.html>

- <http://www.vogella.com/articles/Git/article.html>
- <http://try.github.com/levels/1/challenges/1>
- <https://enterprise.github.com/support>
- <https://support.enterprise.github.com/home>

BOOKS

- Fundamentals of software engineering by Rajib Mall
- Pro C# 2010 and the .NET 4.0 Platform by Andrew Troselen
- C# Programming by Rob Miles

APPENDICES

IDE USED:

VISUAL STUDIO 2010



Microsoft Visual Studio is a powerful IDE that ensures quality code throughout the entire application lifecycle, from design to deployment. Whether we are developing applications for SharePoint, the web, Windows, Windows Phone, and beyond, Visual Studio is the ultimate all-in-one solution. Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other

aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

STANDOUT FEATURES

- User interface built on Windows Presentation Foundation (WPF)
- Improved Start page
- Improved code editor
- Improved IntelliSense
- Call Hierarchy Viewer

WHAT PROBLEMS DOES IT SOLVE?

The newly designed user experience is refreshing for an application showing its age.

The user interface is built on WPF and no longer relies on the limited MDI interface in previous versions; this allows for better multi-monitor support with fly-out windows. The first thing you might notice when opening Visual Studio 2010 is the new Start page. As an xaml file, this page is completely customizable and includes the ability to remove and pin project files in the Recent Projects section.

The code editor has a number of enhancements. You can scale the font by holding down [Ctrl] while scrolling the mouse wheel. In previous versions of Visual Studio, users had to set the font size through a dialog and exit to see if the changes were correct.

In Visual Studio 2010, Box Selection is enhanced to allow for zero-length boxes and improved pasting.

The feature that will see the most use (by accident if not design) is Highlight References. By selecting any symbol, such as a variable or a property, all references to the symbol are highlighted. The symbols can then be navigated by holding down [Ctrl][Shift] and pressing the up/down keys.

IntelliSense has been improved to allow for acronyms based on Pascal casing. For example, typing String.INOE and then a non-alphanumeric character will convert the call to String.IsNullOrEmpty. This still doesn't prevent IntelliSense from interfering when you're writing code that doesn't exist, as you would with a unit test.

The Suggestion Completion mode allows you to type freely without IntelliSense changing the text you typed. You can toggle between Standard and Suggestion Completion modes by pressing [Ctrl][Alt]space.

IntelliSense for JavaScript has seen the most improvement, as it is now able to determine the correct structure of a variable even after the structure is changed.

In the past, I would use .NET Reflector or another tool to analyze a user's call hierarchy; now that functionality is built-in. Right-click the user and choose View Call Hierarchy, and calls to and from the user will be available for browsing.

MATLAB

MATLAB (matrix laboratory) is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

In 2004, MATLAB had around one million users across industry and academia.[2] MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.



Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications.

Windows Presentation Foundation (WPF) provides developers with a unified programming model for building rich Windows smart client user experiences that incorporate UI, media, and documents. Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications. The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typography. WPF is included in the Microsoft .NET Framework, so you can build applications that incorporate other elements of the .NET Framework class library.

The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typography. WPF is included in the Microsoft .NET Framework, so you can build applications that incorporate other elements of the .NET Framework class library.

PROGRAMMING WITH WPF

WPF exists as a subset of .NET Framework types that are for the most part located in the System.Windows namespace. If you have previously built applications with .NET Framework using managed technologies like ASP.NET and Windows Forms, the

fundamental WPF programming experience should be familiar; you instantiate classes, set properties, call methods, and handle events, all using your favorite .NET Framework programming language, such as C# or Visual Basic.

MARKUP & CODE-BEHIND

WPF offers additional programming enhancements for Windows client application development. One obvious enhancement is the ability to develop an application using both markup and code-behind, an experience that ASP.NET developers should be familiar with. You generally use Extensible Application Markup Language (XAML) markup to implement the appearance of an application while using managed programming languages (code-behind) to implement its behavior.

SECURITY

Because XBAPs are hosted in a browser, security is important. In particular, a partial-trust security sandbox is used by XBAPs to enforce restrictions that are less than or equal to the restrictions imposed on HTML-based applications. Furthermore, each HTML feature that is safe to run from XBAPs in partial trust has been tested using a comprehensive security process.

CONTROLS

The user experiences that are delivered by the application model are constructed controls. In WPF, "control" is an umbrella term that applies to a category of WPF classes that are hosted in either a window or a page, have a user interface (UI), and implement some behavior.

WPF CONTROLS BY FUNCTION

The built-in WPF controls are listed here.

- Buttons: Button and RepeatButton.
- Data Display: DataGrid, ListView, and TreeView.
- Date Display and Selection: Calendar and DatePicker.
- Dialog Boxes: OpenFileDialog, PrintDialog, and SaveFileDialog.
- Digital Ink: InkCanvas and InkPresenter.

- Documents: `DocumentViewer`, `FlowDocumentPageViewer`, `FlowDocumentReader`, `FlowDocumentScrollViewer`, and `StickyNoteControl`.
- Input: `TextBox`, `RichTextBox`, and `PasswordBox`.
- Layout: `Border`, `BulletDecorator`, `Canvas`, `DockPanel`, `Expander`, `Grid`, `GridView`, `GridSplitter`, `GroupBox`, `Panel`, `ResizeGrip`, `Separator`, `ScrollBar`, `ScrollViewer`, `StackPanel`, `Thumb`, `Viewbox`, `VirtualizingStackPanel`, `Window`, and `WrapPanel`.
- Media: `Image`, `MediaElement`, and `SoundPlayerAction`.
- Menus: `ContextMenu`, `Menu`, and `ToolBar`.
- Navigation: `Frame`, `Hyperlink`, `Page`, `NavigationWindow`, and `TabControl`.
- Selection: `CheckBox`, `ComboBox`, `ListBox`, `RadioButton`, and `Slider`.
- User Information: `AccessText`, `Label`, `Popup`, `ProgressBar`, `StatusBar`, `TextBlock`, and `ToolTip`.

LAYOUT

When you create a UI, you arrange your controls by location and size to form a layout. A key requirement of any layout is to adapt to changes in window size and display settings. Rather than forcing you to write the code to adapt a layout in these circumstances, WPF provides a first-class, extensible layout system for you.

The cornerstone of the layout system is relative positioning, which increases the ability to adapt to changing window and display conditions. In addition, the layout system manages the negotiation between controls to determine the layout. The negotiation is a two-step process: first, a control tells its parent what location and size it requires; second, the parent tells the control what space it can have.

The layout system is exposed to child controls through base WPF classes. For common layouts such as grids, stacking, and docking, WPF includes several layout controls:

- [Canvas](#) : Child controls provide their own layout.
- [DockPanel](#) : Child controls are aligned to the edges of the panel.
- [Grid](#) : Child controls are positioned by rows and columns.
- [StackPanel](#) : Child controls are stacked either vertically or horizontally.
- [VirtualizingStackPanel](#) : Child controls are virtualized and arranged on a single line that is either horizontally or vertically oriented.
- [WrapPanel](#) : Child controls are positioned in left-to-right order and wrapped to the next line when there are more controls on the current line than space allows.

WPF introduces an extensive, scalable, and flexible set of graphics features that have the following benefits:

- **Resolution-independent and device-independent graphics.** The basic unit of measurement in the WPF graphics system is the device independent pixel, which is 1/96th of an inch, regardless of actual screen resolution, and provides the foundation for resolution-independent and device-independent rendering. Each device-independent pixel automatically scales to match the dots-per-inch (dpi) setting of the system it renders on.
- **Improved precision.** The WPF coordinate system is measured with double-precision floating-point numbers rather than single-precision. Transformations and opacity values are also expressed as double-precision. WPF also supports a wide color gamut (scRGB) and provides integrated support for managing inputs from different color spaces.
- **Advanced graphics and animation support.** WPF simplifies graphics programming by managing animation scenes for you; there is no need to worry about scene processing, rendering loops, and bilinear interpolation. Additionally, WPF provides hit-testing support and full alpha-compositing support.
- **Hardware acceleration.** The WPF graphics system takes advantage of graphics hardware to minimize CPU usage.

EXTENSIBLE APPLICATION MARKUP LANGUAGE (XAML)

XAML stands for Extensible Application Markup Language. It's a simple language based on XML to create and initialize .NET objects with hierarchical relations. Although it was originally invented for WPF it can be used to create any kind of object trees.

Today XAML is used to create user interfaces in WPF, Silverlight, declare workflows in WF and for electronic paper in the XPS standard.

All classes in WPF have parameterless constructors and make excessive usage of properties. That is done to make it perfectly fit for XML languages like XAML.

All you can do in XAML can also be done in code. XAML is just another way to create and initialize objects. You can use WPF without using XAML. It's up to you if you want to declare it in XAML or write it in code. Declare your UI in XAML has some advantages:



- XAML code is short and clear to read
- Separation of designer code and logic
- Graphical design tools like Expression Blend require XAML as source.
- The separation of XAML and UI logic allows it to clearly separate the roles of designer and developer.

PROGRAMMING FRAMEWORK

.NET 4.5



The .NET Framework is a development platform for building apps for Windows, Windows Phone, Windows Server, and Windows Azure. It consists of the common language runtime (CLR) and the .NET Framework class library, which includes classes, interfaces, and value types that support an extensive range of technologies. The .NET Framework provides a managed execution environment, simplified development and deployment, and integration with a variety of programming languages, including Visual Basic and Visual C#.

.NET FRAMEWORK CLASS LIBRARIES

The .NET Framework class library is a library of classes, interfaces, and value types that provide access to system functionality. It is the foundation on which .NET Framework applications, components, and controls are built. The namespaces and namespace categories in the class library are listed in the following table and documented in detail in this reference. The namespaces and categories are listed by usage, with the most frequently used namespaces appearing first.

Namespace	Description
System	The System namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers,

	interfaces, attributes, and processing exceptions.
<u>System.Activities</u>	The System.Activities namespaces contain all the classes necessary to create and work with activities in Window Workflow Foundation.
<u>System.AddIn</u>	The System.AddIn namespaces contain types used to identify, register, activate, and control add-ins, and to allow add-ins to communicate with a host application.
<u>System.CodeDom</u>	The System.CodeDom namespaces contain classes that represent the elements of a source code document and that support the generation and compilation of source code in supported programming languages.
<u>System.Collections</u>	The System.Collections namespaces contain types that define various standard, specialized, and generic collection objects.
<u>System.ComponentModel</u>	The System.ComponentModel namespaces contain types that implement the run-time and design-time behavior of components and controls. Child namespaces support the Managed Extensibility Framework (MEF), provide attribute classes that define metadata for ASP.NET Dynamic Data controls, and contain types that let you define the design-time behavior of components and their user interfaces.
<u>System.Configuration</u>	The System.Configuration namespaces contain types for handling configuration data, such as data in machine or application configuration files. Child namespaces contain types that are used to configure an assembly, to write custom installers for components, and to support a pluggable model for adding functionality to, or removing functionality from, both client and server applications.
<u>System.Data</u>	The System.Data namespaces contain classes for accessing and managing data from diverse sources. The top-level namespace and a number of the child namespaces together form the ADO.NET architecture and

	ADO.NET data providers. For example, providers are available for SQL Server, Oracle, ODBC, and OleDb. Other child namespaces contain classes used by the ADO.NET Entity Data Model (EDM) and by WCF Data Services.
<u>System.Deployment</u>	The System.Deployment namespaces contain types that support deployment of ClickOnce applications.
<u>System.Device.Location</u>	The <u>System.Device.Location</u> namespace allows application developers to easily access the computer's location by using a single API. Location information may come from multiple providers, such as GPS, Wi-Fi triangulation, and cell phone tower triangulation. The <u>System.Device.Location</u> classes provide a single API to encapsulate the multiple location providers on a computer and support seamless prioritization and transitioning between them. As a result, application developers who use this API do not need to tailor applications to specific hardware configurations.
<u>System.Diagnostics</u>	The System.Diagnostics namespaces contain types that enable you to interact with system processes, event logs, and performance counters. Child namespaces contain types to interact with code analysis tools, to support contracts, to extend design-time support for application monitoring and instrumentation, to log event data using the Event Tracing for Windows (ETW) tracing subsystem, to read to and write from event logs and collect performance data, and to read and write debug symbol information.
<u>System.DirectoryServices</u>	The System.DirectoryServices namespaces contain types that provide access to Active Directory from managed code.
<u>System.Drawing</u>	The System.Drawing parent namespace contains types that support basic GDI+ graphics functionality. Child namespaces support advanced two-dimensional and vector graphics functionality, advanced imaging functionality, and print-related and typographical services. A child namespace also contains types that

	extend design-time user-interface logic and drawing.
<u>System.Dynamic</u>	The <u>System.Dynamic</u> namespace provides classes and interfaces that support Dynamic Language Runtime.
<u>System.EnterpriseServices</u>	The System.EnterpriseServices namespaces contain types that define the COM+ services architecture, which provides an infrastructure for enterprise applications. A child namespace supports Compensating Resource Manager (CRM), a COM+ service that enables non-transactional objects to be included in Microsoft Distributed Transaction Coordinator (DTC) transactions. Child namespaces are described briefly in the following table and documented in detail in this reference.
<u>System.Globalization</u>	The <u>System.Globalization</u> namespace contains classes that define culture-related information, including language, country/region, calendars in use, format patterns for dates, currency, and numbers, and sort order for strings. These classes are useful for writing globalized (internationalized) applications. Classes such as <u>StringInfo</u> and <u>TextInfo</u> provide advanced globalization functionalities, including surrogate support and text element processing.
<u>System.IdentityModel</u>	The System.IdentityModel namespaces contain types that are used to provide authentication and authorization for .NET applications.
<u>System.IO</u>	The System.IO namespaces contain types that support input and output, including the ability to read and write data to streams either synchronously or asynchronously, to compress data in streams, to create and use isolated stores, to map files to an application's logical address space, to store multiple data objects in a single container, to communicate using anonymous or named pipes, to implement custom logging, and to handle the flow of data to and from serial ports.
<u>System.Linq</u>	The System.Linq namespaces contain types that support queries that use Language-Integrated Query (LINQ). This includes types that represent queries as objects in

	expression trees.
System.Management	The System.Management namespaces contain types that provide access to management information and management events about the system, devices, and applications instrumented to the Windows Management Instrumentation (WMI) infrastructure. These namespaces also contain types necessary for instrumenting applications so that they expose their management information and events through WMI to potential customers.
System.Media	The System.Media namespace contains classes for playing sound files and accessing sounds provided by the system.
System.Messaging	The System.Messaging namespaces contain types that enable you to connect to, monitor, and administer message queues on the network and to send, receive, or peek messages. A child namespace contains classes that can be used to extend design-time support for messaging classes.
System.Net	The System.Net namespaces contain classes that provide a simple programming interface for a number of network protocols, programmatically access and update configuration settings for the System.Net namespaces, define cache policies for web resources, compose and send e-mail, represent Multipurpose Internet Mail Exchange (MIME) headers, access network traffic data and network address information, and access peer-to-peer networking functionality. Additional child namespaces provide a managed implementation of the Windows Sockets (Winsock) interface and provide access to network streams for secure communications between hosts.
System.Numerics	The System.Numerics namespace contains numeric types that complement the numeric primitives, such as Byte , Double , and Int32 , that are defined by the .NET Framework.

<u>System.Printing</u>	The System.Printing namespaces contain types that support printing, that provide access to the properties of print system objects and enable rapid copying of their property settings to another object of the same type, and that support the interconversion of managed System.PrintTicket objects and unmanaged GDI DEVMODE structures.
<u>System.Reflection</u>	The System.Reflection namespaces contain types that provide a managed view of loaded types, methods, and fields, and that can dynamically create and invoke types. A child namespace contains types that enable a compiler or other tool to emit metadata and Microsoft intermediate language (MSIL).
<u>System.Resources</u>	The System.Resources namespaces contain types that enable developers to create, store, and manage an application's culture-specific resources.
<u>System.Runtime</u>	The System.Runtime namespaces contain types that support an application's interaction with the common language runtime, and types that enable features such as application data caching, advanced exception handling, application activation within application domains, COM interop, distributed applications, serialization and deserialization, and versioning. Additional namespaces enable compiler writers to specify attributes that affect the run-time behavior of the common language runtime, define a contract for reliability between a set of code and other code that takes a dependency on it, and implement a persistence provider for Windows Communication Foundation (WCF).
<u>System.Security</u>	The System.Security namespaces contain classes that represent the .NET Framework security system and permissions. Child namespaces provide types that control access to and audit securable objects, allow authentication, provide cryptographic services, control access to operations and resources based on policy, and support rights management of application-created content.

<u>System.ServiceModel</u>	The System.ServiceModel namespaces contain the types necessary to build Windows Communication Foundation (WCF) service and client applications.
<u>System.ServiceProcess</u>	The System.ServiceProcess namespaces contain types that enable you to implement, install, and control Windows service applications and extend design-time support for Windows service applications.
<u>System.Speech</u>	The System.Speech namespaces contain types that support speech recognition.
<u>System.Text</u>	The System.Text namespaces contain types for character encoding and string manipulation. A child namespace enables you to process text using regular expressions.
<u>System.Threading</u>	The System.Threading namespaces contain types that enable multithreaded programming. A child namespace provides types that simplify the work of writing concurrent and asynchronous code.
<u>System.Timers</u>	The <u>System.Timers</u> namespace provides the <u>Timer</u> component, which allows you to raise an event on a specified interval.
<u>System.Transactions</u>	The System.Transactions namespaces contain types that support transactions with multiple, distributed participants, multiple phase notifications, and durable enlistments. A child namespace contains types that describe the configuration options used by the System.Transactions types.
<u>System.Web</u>	The System.Web namespaces contain types that enable browser/server communication. Child namespaces include types that support ASP.NET forms authentication, application services, data caching on the server, ASP.NET application configuration, dynamic data, HTTP handlers, JSON serialization, incorporating AJAX functionality into ASP.NET, ASP.NET security,

	and web services.
System.Windows	The System.Windows namespaces contain types used in Windows Presentation Foundation (WPF) applications, including animation clients, user interface controls, data binding, and type conversion. System.Windows.Forms and its child namespaces are used for developing Windows Forms applications.
System.Workflow	The System.Workflow namespaces contain types used to develop applications that use Windows Workflow Foundation. These types provide design time and run-time support for rules and activities, to configure, control, host, and debug the workflow runtime engine.
System.Xaml	The System.Xaml namespaces contain types that support parsing and processing the Extensible Application Markup Language (XAML).
System.Xml	The System.Xml namespaces contain types for processing XML. Child namespaces support serialization of XML documents or streams, XSD schemas, XQuery 1.0 and XPath 2.0, and LINQ to XML, which is an in-memory XML programming interface that enables easy modification of XML documents.
Accessibility	The Accessibility and all of its exposed members are part of a managed wrapper for the Component Object Model (COM) accessibility interface.
Microsoft.Activities	The Microsoft.Activities namespaces contain types that support MSBuild and debugger extensions for Windows Workflow Foundation applications.
Microsoft.AspNet.Snapin	The Microsoft.AspNet.Snapin namespace defines the types necessary for the ASP.NET management console application to interact with Microsoft Management Console (MMC). For more information, see "MMC Programmer's Guide" in the MSDN Library .

<u>Microsoft.Build</u>	The Microsoft.Build namespaces contain types that provide programmatic access to, and control of, the MSBuild engine.
<u>Microsoft.CSharp</u>	The Microsoft.CSharp namespaces contain types that support compilation and code generation of source code written in the C# language, and types that support interoperation between the dynamic language runtime (DLR) and C#.
<u>Microsoft.Data.Entity.Build.Tasks</u>	The <u>Microsoft.Data.Entity.Build.Tasks</u> namespace contains two MSBuild tasks that are used by the ADO.NET Entity Data Model Designer (Entity Designer).
<u>Microsoft.JScript</u>	The Microsoft.JScript namespaces contain classes that support compilation and code generation using the JScript language.
<u>Microsoft.SqlServer.Server</u>	The <u>Microsoft.SqlServer.Server</u> namespace contains classes, interfaces, and enumerations that are specific to the integration of the Microsoft .NET Framework common language runtime (CLR) into Microsoft SQL Server, and the SQL Server database engine process execution environment.
<u>Microsoft.VisualBasic</u>	The Microsoft.VisualBasic namespaces contain classes that support compilation and code generation using the Visual Basic language. Child namespaces contain types that provide services to the Visual Basic compiler and types that include support for the Visual Basic application model, the My namespace, lambda expressions, and code conversion.
<u>Microsoft.VisualBasic</u>	The Microsoft.VisualBasic namespaces contain types that support the Visual C++ compiler and types that implement the STL/CLR Library and the generic interface to the STL/CLR Library.

<u>Microsoft.Win32</u>	The Microsoft.Win32 namespaces provide types that handle events raised by the operating system, that manipulate the system registry, and that represent file and operating system handles.
<u>Microsoft.Windows</u>	The Microsoft.Windows namespaces contain types that support themes and preview in Windows Presentation Framework (WPF) applications.
<u>UIAutomationClientsideProviders</u>	Contains a single type that maps client automation providers.
<u>XamlGeneratedNamespace</u>	Contains compiler-generated types that are not intended to be used directly from your code.

DATABASE/BACKEND:

MYSQL



MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history.

The MySQL Community Edition includes:

- Pluggable Storage Engine Architecture
- Multiple Storage Engines: InnoDB , MyISAM, NDB (MySQL Cluster),Memory ,Merge , Archive, CSV
- MySQL Replication to improve application performance and scalability
- MySQL Partitioning to improve performance and management of large database applications
- Stored Procedures to improve developer productivity

DETAILED FEATURES OF MYSQL

The following list shows the most important properties of MySQL. This section is directed to the reader who already has some knowledge of relational databases. We will use some terminology from the relational database world without defining our terms exactly. On the other hand, the explanations should make it possible for database novices to understand to some extent what we are talking about.

Relational Database System: Like almost all other database systems on the market, MySQL is a relational database system.

Client/Server Architecture: MySQL is a client/server system. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc. The clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

Almost all of the familiar large database systems (Oracle, Microsoft SQL Server, etc.) are client/server systems. These are in contrast to the file-server systems, which include Microsoft Access, dBase and FoxPro. The decisive drawback to file-server systems is that when run over a network, they become extremely inefficient as the number of users grows.

SQL compatibility: MySQL supports as its database language -- as its name suggests -- SQL (Structured Query Language). SQL is a standardized language for querying and updating data and for the administration of a database. There are several SQL dialects (about as many as there are database systems). MySQL adheres to the current SQL standard (at the moment SQL:2003), although with significant restrictions and a large number of extensions.

Through the configuration setting `sql-mode` you can make the MySQL server behave for the most part compatibly with various database systems. Among these are IBM DB/2 and Oracle. (The setting `sql-mode` changes some of the syntax conventions, and performs no miracles.

SubSELECTs: Since version 4.1, MySQL is capable of processing a query in the form `SELECT * FROM table1 WHERE x IN (SELECT y FROM table2)` (There are also numerous syntax variants for subSELECTs.)

Views: Put simply, views relate to an SQL query that is viewed as a distinct database object and makes possible a particular view of the database. MySQL has supported views since version 5.0.

Stored procedures: Here we are dealing with SQL code that is stored in the database system.

Stored procedures (SPs for short) are generally used to simplify certain steps, such as inserting or deleting a data record. For client programmers this has the advantage that they do not have to process the tables directly, but can rely on SPs. Like views, SPs help in the administration of large database projects. SPs can also increase efficiency. MySQL has supported SPs since version 5.0.

Triggers: Triggers are SQL commands that are automatically executed by the server in certain database operations (INSERT, UPDATE, and DELETE). MySQL has supported triggers in a limited form from version 5.0, and additional functionality is promised for version 5.1.

Unicode: MySQL has supported all conceivable character sets since version 4.1, including Latin-1, Latin-2, and Unicode (either in the variant UTF8 or UCS2).

User interface: There are a number of convenient user interfaces for administering a MySQL server.

Full-text search: Full-text search simplifies and accelerates the search for words that are located within a text field. If you employ MySQL for storing text (such as in an Internet discussion group), you can use full-text search to implement simply an efficient search function.

Replication: Replication allows the contents of a database to be copied (replicated) onto a number of computers. In practice, this is done for two reasons: to increase protection against system failure (so that if one computer goes down, another can be put into service) and to improve the speed of database queries.

Transactions: In the context of a database system, a transaction means the execution of several database operations as a block. The database system ensures that either all of the operations are correctly executed or none of them. This holds even if in the middle of a transaction there is a power failure, the computer crashes, or some other disaster occurs. Thus, for example, it cannot occur that a sum of money is withdrawn from account A but fails to be deposited in account B due to some type of system error.

Transactions also give programmers the possibility of interrupting a series of already executed commands (a sort of revocation). In many situations this leads to a considerable simplification of the programming process. In spite of popular opinion, MySQL has supported transactions for a long time. One should note here that MySQL can store tables in a variety of formats. The default table format is called MyISAM, and this format does not support transactions. But there are a number of additional formats that do support transactions. The most popular of these is InnoDB, which will be described extensively in this book.

Foreign key constraints: These are rules that ensure that there are no cross references in linked tables that lead to nowhere. MySQL supports foreign key constraints for InnoDB tables.

GIS functions: Since version 4.1, MySQL has supported the storing and processing of two-dimensional geographical data. Thus MySQL is well suited for GIS (geographic information systems) applications.

Programming languages: There are quite a number of APIs (application programming interfaces) and libraries for the development of MySQL applications. For client programming you can use, among others, the languages C, C++, Java, Perl, PHP, Python, and Tcl.

ODBC: MySQL supports the ODBC interface [Connector/ODBC](#). This allows MySQL to be addressed by all the usual programming languages that run under Microsoft Windows (Delphi, Visual Basic, etc.). The ODBC interface can also be implemented under Unix, though that is seldom necessary.

Windows programmers who have migrated to Microsoft's new .NET platform can, if they wish, use the ODBC provider or the .NET interface Connector/.NET.

Platform independence: It is not only client applications that run under a variety of operating systems; MySQL itself (that is, the server) can be executed under a number of operating systems. The most important are Apple Macintosh OS X, Linux, Microsoft Windows, and the countless Unix variants, such as AIX, BSDI, FreeBSD, HP-UX, OpenBSD, Net BSD, SGI Iris, and Sun Solaris.

Speed: MySQL is considered a very fast database program. This speed has been backed up by a large number of benchmark.

IDE FOR DATABASE

MYSQL WORKBENCH



MySQL Workbench is a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system. It is the successor to DBDesigner 4 from fabFORCE.net, and replaces the previous package of software, MySQL GUI Tools Bundle. [MySQL Workbench](#) enables a DBA, developer, or data architect to visually design, generate, and manage all types of databases including Web, OLTP, and data warehouse databases. It includes everything a data modeler needs for creating complex ER models, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort. MySQL Workbench is available on Windows, Linux and Mac OS.

BENEFITS

- Simplifies database design and maintenance
- Automates time-consuming and error-prone tasks
- Enables data architects to visualize requirements, communicate with stakeholders, and resolve design issues before a major investment of time and resources is made
- Enables model-driven database design—the most efficient methodology for creating valid and well-performing databases—while providing the flexibility to respond to evolving business requirements
- Provides capabilities to forward-engineer physical database designs and reverse-engineer existing databases
- Allows you to import SQL scripts to build models and export models to DDL scripts that can be run at a later time
- Enables you to compare two live databases or a model and a live database, visually see the differences, and perform a synchronization between a model and a live database or vice versa

- Simplifies the documentation of database designs, providing a point-and-click process that delivers documentation in HTML or plain-text format

TOOLS

The three main tools of MySQL Workbench are:

- SQL Development
- Data Modelling
- Server Administration

PROGRAMMING LANGUAGE

C# - C SHARP



C# is a type-safe, object-oriented language that is simple yet powerful, allowing programmers to build a breadth of applications. C# is a multi-paradigm programming language encompassing imperative, declarative, functional, generic, object-oriented(class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure. C# is intended to be a simple, modern, general-purpose, object-oriented programming language.

CHARACTERISTICS OF C#:

C# was developed to bring rapid development to C++ without sacrificing the power and control of C and C++. C# provides various characteristics, which are:

Simple:

C# eliminates the use of tedious operators such as -->, and pointers. C# treats `int` and `Boolean` as two different data types, which enable the compiler to recognize the use of `=` in place of `==` with `if` statement.

Consistent:-

C# supports only one integer type and there is no limitation of range.

Modern:-

C# contains various features necessary to develop web applications. Following are the features of C#:

It provides automatic garbage collection.

It provides robust security model.

It provides decimal data type for financial application.

It provides modern approach for debugging.

It provides a rich intrinsic model for error handling.

Object Oriented:-

C# supports all the features of object oriented language such as encapsulation, inheritance and polymorphism. It treats everything as an object and there are no global

functions, variables and constants in C#.

Type Safe:-

C# provides various type safe measures, which are:

Dynamically allocated objects and arrays are initialized to zero.

Products an error message while using an uninitialized variable.

Checks the range of an array and warns when the access goes out of bound.

Unsafe casts are not allowed.

Enforces overflow checking in arithmetic operations.

Versionable:-

C# supports versioning that enables the existing applications to run on different

versions with the help of new and override command.

Compatible:

C# contains the .NET specifications and therefore, allows inter operation with other .NET languages.

Flexible:-

C# does not support pointers but you may use pointers to manipulate the data of certain classes and methods by declaring them unsafe.

Inter-operability:

C# enables a program to call out any native API. It also allows the use of COM objects written in different languages.

MATLAB

MATLAB® is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java™.



You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing.

DIA FOR DIAGRAM DRAWING & MODELING

Dia is free and open source general-purpose diagramming software, developed as part of the GNOME project's office suite and was originally created by Alexander Larsson. Dia uses a controlled single document interface (CSDI) similar to GIMP and Sodipodi.

Dia has a modular design with several shape packages available for different needs: flowchart, network diagrams, circuit diagrams, and more. It does not restrict symbols and connectors from various categories from being placed together.

Dia is a gtk+ based diagram creation program released under the GPL license.

Dia is inspired by the commercial Windows program 'Visio', though more geared towards informal diagrams for casual use. It can be used to draw many different kinds of diagrams. It currently has special objects to help draw entity relationship diagrams, UML diagrams, flowcharts, network diagrams, and many other diagrams. It is also possible to add support for new shapes by writing simple XML files, using a subset of SVG to draw the shape.

It can load and save diagrams to a custom XML format (gzipped by default, to save space), can export diagrams to a number of formats, including EPS, SVG, XFIG, WMF and PNG, and can print diagrams (including ones that span multiple pages).

GOOGLE SPREADSHEET INTERFACE:

With Google Spreadsheets, we can easily create, share, and edit spreadsheets online. Here are a few specific things we can do:

- *Import and export these file types: .xls, .csv, .txt and .ods. We can also export data to a PDF or an HTML file.*

- *Format cells and edit formulas so we can calculate results and make data look the way we want it.*
- *Chat in real time with others who are editing our spreadsheet.*
- *Embed a spreadsheet, or a section of a spreadsheet, in our blog or website.*

CACOO:: ONLINE DRAWING TOOL



Cacoo is a diagram creation tool that runs in your web browser. Multiple people can work together on the same diagram in real time. Diagrams can be published directly to websites, wikis, and blogs.

CREATING DIAGRAMS

- Elements can be dragged and drop to easily create diagrams.
- Elements can be linked together with connectors.
- Connectors automatically move when elements are repositioned.
- You can use a text box and put text anywhere you like.
- You can upload images from your PC and include them in Diagrams.
- You can take screenshots of your computer from within Cacoo.
- Smart styles can easily be applied to stencils.
- You can have multiple sheets in a diagram and use them as backgrounds or layers.
- When you move the objects on your canvas, they will be snapped at the objects or grids nearby and align automatically.
- Copying, pasting and other functionality of basic drawing software is also built in to Cacoo.
- All actions are stored so there are unlimited levels of undo.
- You can import an image from the other websites by indicating the URL.
- The imported image can be easily trimmed only using your mouse.
- According to your editing status, tips will be shown on the right bottom corner of the canvas.

COLLABORATION

- You can invite collaborators to work with you in Cacoo.
- Multiple people can edit a diagram in real time.
- There is a chat function in the editor so people can communicate while creating diagrams.
- People can leave comments about the diagrams.
- Each user can set their own user icon.
- When editing with multiple people, users icons appear on selected objects.
- Sharing diagrams become much smoother. Diagrams in the shared folders can be accessible and editable by people who you have shared the folder with.

SHARING DIAGRAMS

- If you keep the diagram private then other users can't see it.
- If you make the diagram URL public, then anyone who knows the URL can see it.
- Publishing a diagram to a blog can be useful in various ways.
- You can place code into blogs to create a slideshow
- Published images always display the most recent version.
- Diagrams can be exported to SVG format (Plus Plan users only) and PNG format. (More formats will be available in the future.)
- Diagrams can be posted to Twitter/Facebook/GoogleBuzz
- Diagrams can be displayed in SVG format for printing. (Plus Plan users only. A few browsers are not supported.)

MANAGING DIAGRAMS

- Diagrams can be placed into folders.
- Diagrams can be copied.
- Diagrams can be displayed as thumbnails or as a list.

LANGUAGES AND TIME ZONES

- All pages and notification e-mails support English and Japanese
- Users can enter text from almost all languages.
- Dates are displayed relative to your local time zone.

SECURITY

- Private diagrams can only be seen by users you select.
- URLs which you do not share can not be found by other users or search engines.
- All editing and management is protected by SSL.
- In order to access information about diagrams a Cacoo ID and password are required.
- User passwords are encrypted on Cacoo's server.

API

- You can access Cacoo using the API.
- The Cacoo API supports OAuth and an API Key.

By using the Cacoo API you are able to interact with Cacoo from other services and applications.

Authorization Methods

There are two ways to access the Cacoo API.

1. API KEY

The API key allows you make requests to the Cacoo API. You can make an API key here.

API KEY

Append your API key to requests to the API to return data from your account.(Parameter name "apiKey")

Example: <https://cacoo.com/api/v1/diagrams.json?apiKey=abcdefghijklmn>

2. OAUTH

OAuth 1.0a is supported as an authorization method for Cacao. You can register applications [here](#).

You can get your Access Token from the following links.

APPLICATIONS

Access Token:https://cacao.com/oauth/access_token

Authorize:<https://cacao.com/oauth/authorize>

Request Token:https://cacao.com/oauth/request_token

VERSION CONTROL SYSTEM : GITHUB



GitHub is a web-based hosting service for software development projects that use the Git revision control system. GitHub offers both paid plans for private repositories, and free accounts for open source projects. As of May 2011, GitHub was the most popular open source code repository site. GitHub Inc. was founded in 2008 and is based in San Francisco, California.

DESCRIPTION

The site provides social networking functionality such as feeds, followers and the network graph to display how developers work on their versions of a repository.

GitHub also operates other services: a pastebin-style site called Gist that provides wikis for individual repositories and web pages that can be edited through a Git repository, a slide hosting service called Speaker Deck, and a web analytics platform called Gauges.

As of January 2010, GitHub is operated under the name GitHub, Inc.

The software that runs GitHub was written using Ruby on Rails and Erlang by GitHub, Inc. (previously known as Logical Awesome) developers Chris Wanstrath, PJ Hyett, and Tom Preston-Werner.

LIMITATIONS AND CONSTRAINTS

According to the terms of service, if an account's bandwidth usage significantly exceeds the average of other GitHub customers, the account's file hosting service may be immediately disabled or throttled until bandwidth consumption is reduced. In addition, while there is no hard limit, the guideline for the maximum size of a repository is one gigabyte.

GLOSSARY.

FRS Face Recognizer System

Apps Application

SRS Software Requirement Specification

DFD Data Flow Diagram

ERD Entity Relationship Diagram

GUI Graphical User Interface

UI User Interface

DB Database

COCOMO Constructive Cost Model

SDK Sweater Development Kit

WPF Windows Presentation Framework

XAML Extensible application Markup Language

IDE Integrated Development Environment

HTML Hyper Text Markup Language

www World Wide Web

DBMS Database Management System

Sync Synchronization

cs C Sharp

KLOC Estimated size of the software product expressed in Kilo

Tdev Estimated time to develop the software, expressed in months.

Effort Total effort required to develop the software product, expressed in person-month (PM).

PM Person-month