INDEX

1. Introduction 2
  1.1 Purpose of the Document 2
  1.2 Scope 2
  1.3 Design Constraints and Limitations 3
    1.3.1 Constraint 3
    1.3.2 Limitations 3
  1.4 Goals and Objectives 3
2. Data Design 4
  2.1 Data Objects 4
  2.2 ER Diagrams 8
  2.3 Data Dictionary 10
  2.4 Internal Software Data Structures 15
  2.5 Database Description 20
  2.6 Database Normalization 20
3. Architectural and Component-level Design 21
  3.1 Structure Chart 21
  3.2 Editor Module 21
  3.3 Project Module 21
  3.4 Debugger Module 21
  3.5 Database Module 22
4. User Interface Design 22
  4.1 Menus 23
    4.1.1 File Menus 23
    4.1.2 Edit Menu 24
    4.1.3 Insert Menu 24
    4.1.4 Database Menu 26
    4.1.5 Run Menu 26
    4.1.6 Help Menu 26
  4.2 Tabs 26
    4.2.1 File Tab 26
    4.2.2 Code Tab 27
    4.2.3 Inspector Tab 27
    4.2.4 Properties Tab 28
    4.2.5Events Tab 28
    4.2.6 Database Tab 29
  4.3 Main Panel 29
5. System Modeling 30
  5.1 DFD 30
    5.1.1 DFD Level 0 30
    5.1.2 DFD Level 1 31
  5.2 Use Case Diagram 32
6.Class Diagram 36
7. Project Schedule 38
  7.1 Gannt Chart 38

1. INTRODUCTION

1.1 Purpose of the Document

This document is the Initial Design Report for our project regarding the Ajax IDE software. During the preparation of the report the main purpose is to realize and overcome design issues and come up with appropriate solutions. This document aims to establish a basis for the soon-to-be prepared Final Design Report and therefore tries to give a clear understanding of the design of the software.
In the preparation of the document we found the inclusion of the following necessary :

- Modular Specifications
- UML Diagrams
- Screenshots
- Updated Schedule

With the information provided in this document it is aimed to get a clearer understanding of the software before the final design phase and detect the shortcomings beforehand in case they exist.

1.2 Scope

Ajax is mainly a web development technique for creating interactive web applications. The components that made up Ajax were being used even before the naming took place. Due to the convenience that Ajax brings to developers it has gained popularity quite quickly. In this project we will design and implement a Graphical Development Environment for webpages with Ajax which helps the developers further by putting many options and features together.

The software is to include an editor to write and edit the neccesary codes. The text-highlighting method will be used to increase readability. Predefined scripts and actions will be available. Features such as adding, locating, editing, removing scripts and actions will be put to use. Without using any other software, debugging the code when necessary will also be at hand. The user will be able to see the effect of the changes made and run the code. The advanced graphical interface is to be implemented as a way of easing the processes of the user. Hence, the user of the software is to be satisfied with every need he/she has through one complete package.

1.3 Design Constraints and Limitations

1.3.1 Constraints

System: Our software should be secure as there will be database connections and editing with transfers of username, password and potentially other critical data.
Different kinds of databases will be usable so the software should offer such adaptability.
There will be projects with many files and structures within to be used at the same time. This should not be overlooked.

Interface: The interface should be user-friendly and easy to understand as well as to use. Since the user’s main goal is to do the work more easily with tools and other components available, the graphical user interface should be ready to face such demands.

1.3.2 Limitations

Time: Time is an important limitation as there are many other duties of the project members and the time is strictly defined as 9 months. Scheduling and proper preparation is of the essence.
Employee Skills: The skills of the project members are developing as the project continues. While building the software the members are also occupied with learning the processes to be involved.
Hardware: .....
Portability: The Windows XP will be used during implementation.
Programming Language: Java programming language has been chosen after the meetings mentioned in the previous report. The libraries of Java Language and the skill level of the members on Java contribute to some limitations.

1.4 Goals and Objectives

During the analysis of the software we have focused on the following goals and objectives. These will be examined again as non-functional requirements in the following sections of the report as they also constitute an important part of the requirements of our project.

Easy to Understand : The software package is bound to include many help options and documentary to help the user understand how to use the software more efficiently with less effort.

Easy to Use : One of the most important reasons for a developer to use such a software is to make his life easier with the features presented through the software which are not available in a regular editor. Consequently, a great deal of effort is to be made to make the software satisfactory to these needs : Text-Highlighting, pre-defined scripts, etc...

Performance : Since such software should satisfy complex developing as well as simple ones, the performance issue is rather an important one. The aimed-customer database is also making the performance goals more important since it is clearly more obvious and critical to someone who is a developer than a regular software user.

Update Readiness : The technology that the software will depend on in terms of scripts , actions, debugging and so on, makes the update issue an important one. One of the main goals of the project is to make sure not to overlook this concept as with time and with the high speed of the introduction of new features it should be easy to add new modules, remove old ones which will not be necessary anymore.

2. DATA DESIGN

In database, project, project files, their codes and its members, files' inspectors, their form elements, properties and events of form elements and database which is used at project are stored at database. In order to store the data in a structured form, the data objects will be used. In this section, we will look at the data objects, their relationships, the ER-diagram and the data dictionary to describe the data.

2.1 Data Objects

Project
Project entity is stored the data of projects of the program. When the user opens a new project, the data of project is stored. The attributes of entity are:

- Project_ID
- Project_Name

Project_ID will be integer and it will be unique for each project. Also Project_Name will be unique for each project and it will be string.

File

File entity will stored the files of project which has relation with Inspector entity and Code entity.

- File_ID
- File_Name
- Project_ID

File_ID will be integer and it will be primary key. The File_Name will be string and Project_ID will be foreign key and it references the relation with File entity and Project entity. Besides, File_Name and Project_ID together will be unique.

Inspector

Inspector entity will be investigate the File entity and also it has relation with Form_Elements entity.

- Inspector_ID
- Form_Name
- File_ID

Inspector_ID will be integer and it will be primary key. Form_Name will be string and it will store the name of forms in each inspector. File_ID will be foreign key and it reference the relation with File Entity. Moreover, File_ID and Forn_Name together will be unique.

Form_Elements

Form_Elements entity will stored elements of each form. The attributes are:

- Form_Elements_ID
- Inspector_ID
- Form_Elements_Name
- Line_Number

Form_Elements_ID will be integer and primary key. Inspector_ID will reference between Form_Elements entity and Inspector_Elements entity. Form_Elements_Name will be string and stored the name of elements. Line_Number will be integer and show the line number of each element. Inspector_ID and Form_Elements_Name together will be unique.

Events

Events entity will store the event of form elements. The attributes are:

- Events_ID
- On_Abort
- On_Blur
- On_Change
- On_Clik
- On_Dbclick
- On_Error
- On_Focus
- On_Keydown
- On_Keypress
- On_Keyup
- On_Load
- On_Mousdown
- On_Mousemove
- On_Mouseup
- On_Mouseover
- On_Mouseout
- On_Reset
- On_Resize

Events_ID will be primary key. All the other attributes will be true-false.

Properties

Properties entity will store the properties of form elements. The attributes are:

- Property_ID
- Property_Name
- Border_Color
- Border_Style
- Color
- Font_Name
- Font_Color
- Font_Size
- Font_Style

Porperty_ID will be primary key.

Code

Code entity will store the code data of file. The attributes are:

- Code_ID
- Code_Name
- Code_Type
- Code_Line_Number
- File_ID

Code_ID will be integer and will be primary key. Code_Name will be string. Code_Type is 1 or 2 which means that 1shows that code type is class and 2 shows that type is function. Code_Line_Number will be integer and stores the line number. File_ID is foreign key and reference to File entity. File_ID, Code_Name and Code_Type together will be unique.

Members

Members entity will store the variables and the functions of the each class. The attributes are:

- Member_ID
- Code_ID
- Member_Name
- Member_Line_Number

Member_ID will be integer and it will be primary key. Code_ID is foreign key and refernce to Code Entity. Member_Name will be string. Code_ID and Member_Name together will be unique. Member_Line_Number will be string and store line number of member.

Database

Database entity will be stored the data of query which the user occur when they will connect to server. The attributes are:

- Database_ID
- Database_Name
- Project_ID

Database_ID will be integer and it will be primary key. Database_Name is string and store the name of database which the user connect. Porject_ID is foreign key and reference Project entity.

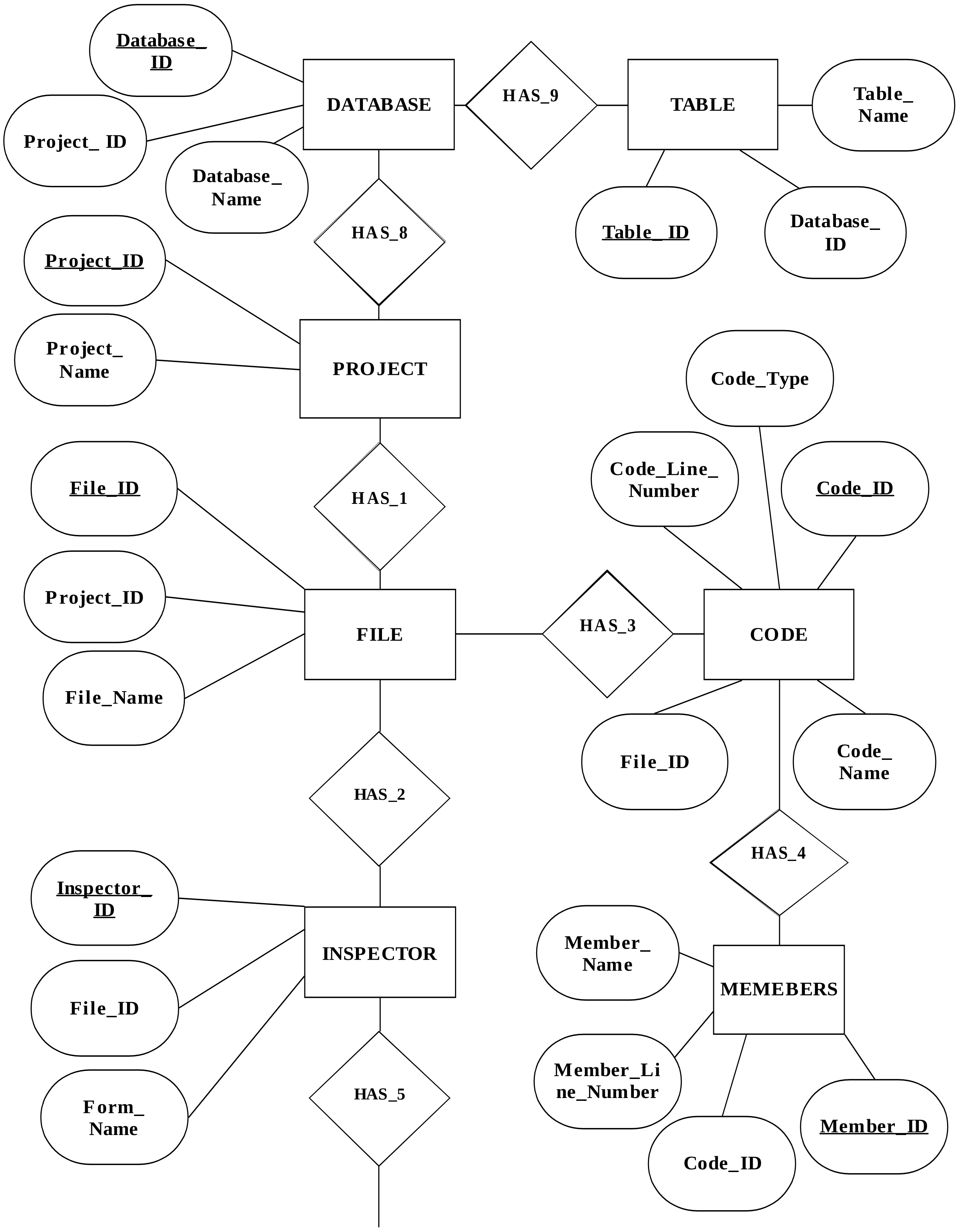
Table

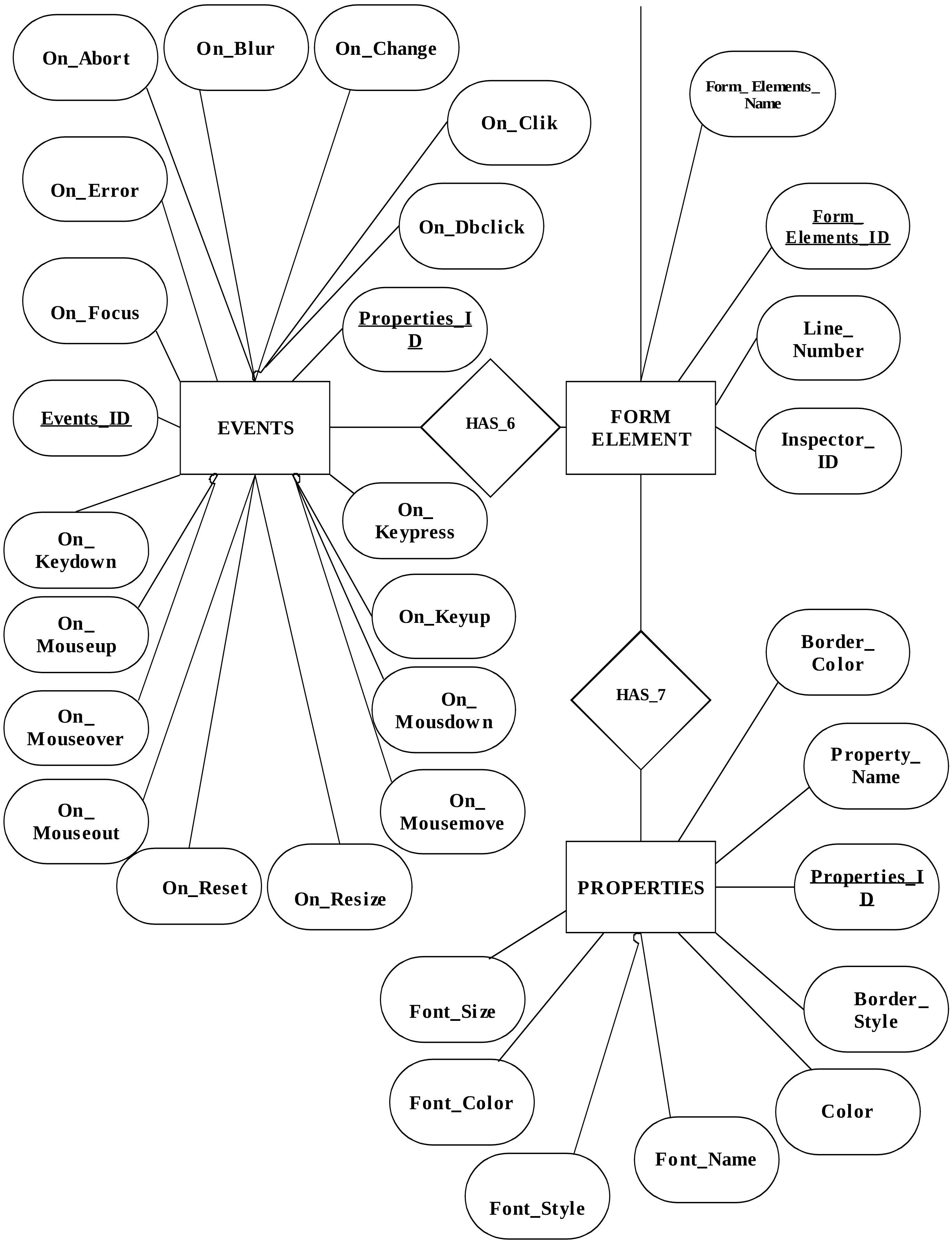
Table entity will stored the tables according to user query. The attributes are:

- Table_ID
- Tabale_Name
- Database_ID

Table_ID will be integer and primary key. Table_Name will be string and stored the name of table. Database_ID will be foreign key and reference to Database entity.

2.2 ER Diagrams




On_Abort
On_Blur
On_Change
On_Clik
On_Error
On_Dbclick
On_Focus
Properties_ID
Events_ID
EVENTS
HAS_6
FORM ELEMENT
Form_ Elements_ Name
Form_ Elements_ID
Line_ Number
Inspector_ ID
On_ Keydown
On_ Keypress
On_ Mouseup
On_Keyup
On_ Mouseover
On_ Mousdown
On_ Mouseout
On_ Mousemove
On_Reset
On_Resize
HAS_7
Border_ Color
Property_ Name
PROPERTIES
Properties_ID
Font_Size
Border_ Style
Font_Color
Color
Font_Name
Font_Style

2.3 Data Dictionary

Project

Name	PROJECT
Alias	-
Where / How used	A new project which users open or load.
Description	When the user opens a new project, the data of project is stored.

Project_ID

Name	Project_ID
Alias	-
Where / How used	It is assigned all projects when they are created.
Description	Each project has a unique Project_ID.

Project_Name

Name	Project_Name
Alias	-
Where / How used	It is assigned all projects when they are created.
Description	Each project has a unique Project_Name.

FILE

Name	FILE
Alias	-
Where / How used	While creating new file or loading a file.
Description	File entity will stored the files of project.

File_ID

Name	File_ID
Alias	-
Where / How used	It is assigned all files when they are created.
Description	Each file has a unique File_ID.

File_Name

Name	File_Name
Alias	-
Where / How used	It is assigned all files when they are created.
Description	Each file has a unique File_Name.

INSPECTOR

Name	INSPECTOR
Alias	-
Where / How used	Every file entity has inspector.
Description	Inspector entity is investigating the File entity.Form names saved in it.

Inspector_ID

Name	Inspector_ID
Alias	-
Where / How used	Primary key of inspector.
Description	Each inspector has a unique Inspector_ID.

Form_Name

Name	Form_Name
Alias	-
Where / How used	It is saved in INSPECTOR entity.
Description	Name of the form.

FORM ELEMENT

Name	FORM ELEMENT
Alias	-
Where / How used	When INSPECTOR takes action.
Description	Each form and their line number stored in it.It has also form elements name

Form_Elements_ID

Name	Form_Elements_ID
Alias	-
Where / How used	Primary key of FORM ELEMENT.
Description	Each FORM ELEMENT has a unique Form_Elements_ID

Form_Elements_Name

Name	Form_Elements_Name
Alias	-
Where / How used	Member of FORM ELEMENTS
Description	It is string and stores the name of elements

Line_Number

Name	Line_Number
Alias	-
Where / How used	Member of FORM ELEMENTS
Description	It is integer and showa the line number of each element.

EVENTS

Name	EVENTS
Alias	-
Where / How used	When FORM ELEMENTS takes action.
Description	Events entity will store the event of form elements. These attributes are: abort,blur,change,click,double,click,error,focus,keydown,keypress,keyup, mouse down, mouse up, mouse over, mouse out, reset and resize.

Events_ID

Name	Events_ID
Alias	-
Where / How used	Primary key of Events_ID.
Description	Each EVENTS has a unique Events_ID

PROPERTIES

Name	PROPERTIES
Alias	-
Where / How used	When FORM ELEMENTS takes action.
Description	It stores the properties of form elements like name, border color, border style, color, font name, font color, font size, font style

Porperty_ID

Name	Porperty_ID
Alias	-
Where / How used	Primary key of PROPERTIES.
Description	Each PROPERTIES has a unique Porperty_ID.

CODE

Name	CODE
Alias	-
Where / How used	Every file entity has CODE entity.
Description	It stores the code data of file.

Code_ID

Name	Code_ID
Alias	-
Where / How used	Primary key of CODE.
Description	Each CODE has a unique Code_ID.

Code_Name

Name	Code_Name
Alias	-
Where / How used	Member of CODE entity
Description	It is name of the classes or functions in code. I is unique

Code_Type

Name	Code_Type
Alias	-
Where / How used	Member of CODE entity
Description	It is an integer:1 or 2.1 is used for classes 2 is used functions.I is unique

Code_Line_Number

Name	Code_Line_Number
Alias	-
Where / How used	Member of CODE entity
Description	It is line number of code.

MEMBERS

Name	MEMBERS
Alias	-
Where / How used	Every CODE entity has MEMBERS.
Description	It stores the variables and the functions of the each class. It has unique Code_ID

Member_ID

Name	Member_ID
Alias	-
Where / How used	Primary key of MEMBERS.
Description	Each MEMBERS has a unique Member_ID.

Member_Name

Name	Member_Name
Alias	-
Where / How used	Element of MEMBERS.
Description	Name of elements.It is unique.

Member_Line_Number

Name	Member_Line_Number
Alias	-
Where / How used	Element of MEMBERS.
Description	It shows the line number of member.

DATABASE

Name	DATABASE
Alias	-
Where / How used	Creating when user will connect to server
Description	It stores the data of query

Database_ID

Name	Database_ID
Alias	-
Where / How used	Primary key of DATABASE.
Description	Each DATABASE has a unique Database_ID.

Database_Name

Name	Database_Name
Alias	-
Where / How used	Element of MEMBERS.
Description	It is string and store the name of database which the user connect.

TABLE

Name	TABLE
Alias	-
Where / How used	When DATABASE takes action
Description	It stores the tables according to user query

Table_ID

Name	Table_ID
Alias	-
Where / How used	Primary key of TABLE
Description	Each TABLE has a unique Table_ID.

Table_Name

Name	Table_Name
Alias	-
Where / How used	Element of TABLE
Description	It is string and store the name of the table.

2.4 Internal Software Data Structure

Porject

```
CREATE TABLE Project
(
Porject_ID INTEGER,
Project_Name VARCHAR(32),
PRIMARY KEY(Project_ID),
);
```

Data	Type&Size	Format
Porject_ID	INTEGER	Number
Project_Name	VARCHAR	Text

File

```
CREATE TABLE File
(
File_ID INTEGER,
Porject_ID INTEGER,
File_Name VARCHAR(32),
PRIMARY KEY(File_ID),
FOREIGN KEY(Project_ID) REFERENCES Project,
UNIQUE (Project_ID, File_Name),
);
```

Data	Type&Size	Format
File_ID	INTEGER	Number
File_Name	VARCHAR	Text
Porject_ID	INTEGER	Number

Inspector

```
CREATE TABLE Inspector
(
Inspector_ID INTEGER,
Form_Name VARCHAR(32),
File_ID INTEGER,

PRIMARY KEY(Inspector_ID),
FOREIGN KEY(File_ID ) REFERENCES File,
UNIQUE (File_ID, Form_Name),
);
```

Data	Type&Size	Format
Inspector_ID	INTEGER	Number
Form_Name	VARCHAR	Text
File_ID	INTEGER	Number

Form_Elements

```
CREATE TABLE Form_Elements
(
Form_Elements_ID INTEGER,
Form_Elements _Name VARCHAR(32),
Line_Number INTEGER,
Inspector_ID INTEGER,
PRIMARY KEY(Form_Elements_ID),
FOREIGN KEY(Inspector_ID) REFERENCES Inspector,
UNIQUE (Inspector_ID, Form_Elements_Name),
);
```

Data	**Type&Size**	**Format**
Form_Elements_ID	INTEGER	Number
Form_Elements _Name	VARCHAR	Text
Line_Number	INTEGER	Number
Inspector_ID	INTEGER	Number

Events

```
CREATE TABLE Events
(
Events_ID INTEGER,
On_Abort INTEGER,
On_Blur INTEGER,
On_Change INTEGER,
On_Clik INTEGER,
On_Dbclick INTEGER,
On_Error INTEGER,
On_Focus INTEGER,
On_Keydown INTEGER,
On_Keypress INTEGER,
On_Keyup INTEGER,
On_Load INTEGER,
On_Mousdown INTEGER,
On_Mousemove INTEGER,
On_Mouseup INTEGER,
On_Mouseover INTEGER,
On_Mouseout INTEGER,
On_Reset INTEGER,
On_Resize INTEGER,
);
```

Data	Type&Size	Format
Events_ID	INTEGER	Number
On_Abort	TRUE/FALSE	TRUE/FALSE
On_Blur	TRUE/FALSE	TRUE/FALSE
On_Change	TRUE/FALSE	TRUE/FALSE
On_Change	TRUE/FALSE	TRUE/FALSE
On_Clik	TRUE/FALSE	TRUE/FALSE
On_Dbclick	TRUE/FALSE	TRUE/FALSE
On_Error	TRUE/FALSE	TRUE/FALSE
On_Focus	TRUE/FALSE	TRUE/FALSE
On_Keydown	TRUE/FALSE	TRUE/FALSE
On_Keypress	TRUE/FALSE	TRUE/FALSE
On_Keyup	TRUE/FALSE	TRUE/FALSE
On_Load	TRUE/FALSE	TRUE/FALSE
On_Mousdown	TRUE/FALSE	TRUE/FALSE
On_Mousemove	TRUE/FALSE	TRUE/FALSE
On_Mouseup	TRUE/FALSE	TRUE/FALSE
On_Mouseover	TRUE/FALSE	TRUE/FALSE
On_Mouseout	TRUE/FALSE	TRUE/FALSE
On_Reset	TRUE/FALSE	TRUE/FALSE
On_Resize		

Properties

```
CREATE TABLE Properties
(
Properties_ID INTEGER,
Property_Name VARCHAR,
Border_Color INTEGER,
Border_Style INTEGER,
Color INTEGER,
Font_Name VARCHAR,
Font_Color INTEGER,
Font_Size INTEGER,
Font_Style INTEGER,
);
```

Data	Type&Size	Format
Properties_ID INTEGER,	INTEGER	Number
Property_NameVARCHAR,	VARCHAR	Text
Border_Color INTEGER,	INTEGER	Number
Border_Style INTEGER,	INTEGER	Number
Color INTEGER,	INTEGER	Number
Font_Name INTEGER,	VARCHAR	Text
Font_Color INTEGER,	INTEGER	Number
Font_Size INTEGER,	INTEGER	Number
Font_Style INTEGER,	INTEGER	Number

Code

```
CREATE TABLE Code
(
Code_ID INTEGER,
Code _Name VARCHAR(32),
Code_Type INTEGER,
Code_Line_Number INTEGER,
File_ID INTEGER,
PRIMARY KEY(Code_ID),
FOREIGN KEY(File) REFERENCES File,
UNIQUE (Code_ID, Code_Name,Code_Type),
);
```

Data	**Type&Size**	**Format**
Code_ID	INTEGER	Number
Code _Name	VARCHAR	Text
Code_Line_Number	INTEGER	Number
Code_Type	INTEGER	Number
File_ID	INTEGER	Number

Members

```
CREATE TABLE Code
(
Member_ID INTEGER,
Member _Name VARCHAR(32),
Member_Line_Number INTEGER,
Code_ID INTEGER,
PRIMARY KEY(Code_ID),
FOREIGN KEY(Code_ID) REFERENCES Code,
UNIQUE (Code_ID, Member_Name),
);
```

Data	**Type&Size**	**Format**
Member_ID	INTEGER	Number
Member _Name	VARCHAR	Text
Member_Line_Number	INTEGER	Number
Code_ID	INTEGER	Number

Database

```
CREATE TABLE File
(
Database_ID INTEGER,
Porject_ID INTEGER,
Project_Name VARCHAR(32),
PRIMARY KEY(Database_ID),
FOREIGN KEY(Project_ID) REFERENCES Project,
);
```

Data	Type&Size	Format
Database_ID	INTEGER	Number
Database_Name	VARCHAR	Text
Porject_ID	INTEGER	Number

Table

```
CREATE TABLE File
(
Table_ID INTEGER,
Project_Name VARCHAR(32),
Database_ID INTEGER,
PRIMARY KEY(Table_ID),
Database KEY(Project_ID) REFERENCES Project,
);
```

Data	Type&Size	Format
Table_ID	INTEGER	Number
Table_Name	VARCHAR	Text
Database_ID	INTEGER	Number

2.5 Database Description

Database will store all of the information of the project which will be occurred by the user. Also, the File, Code, Inspectors, Database relation are stored in the database. When the system needs retrieving data, SQL queries are used to get the necessary records.

2.6 Database Normalization

The database in our software is designed avoiding redundancy cases and we tried to suit them to the BCNF notation. To obey these rules we did some modifications over the real data

tables. Instead of creating separate tables for each relation, we added a new attribute to one of the entities of the relation that is, a foreign key and reference to the other table. Important modification is assigning a ID of one entity to the entity which is connect to it. This was done to simplify the interaction between these tables. By this modification, we avoided redundancy of the tables. As a result, there are no insertion, update and deletion
anomalies. Moreover, these will ease the queries for relations.

3. ARCHITECTURAL and COMPONENT-LEVEL DESIGN

One of the main concepts of developing a software is a well-done architectural and componet-level design. A proper design will lead success in software and make the job more obvious and easier.

3.1 Structure Chart

Structure Chart is composed of modules of the software as Editor Module, Project Module, Debugger Module and finally Database Module. All these modules has a relationship and none of them is independently working. Especially Editor Module is core of the software and other modules are working mainly in relation with that module.

3.2 Editor Module

Editor Module is said to be core of all modules that is obvious to and mostly used by developer. Editor Module is composed of two main parts named as Code part and Design part. These code and design parts are to be thought as together, because a change in one of these parts will directly affect the other part. Code part of the module will include some properties which are special to itself as predefined code segments for code language that developer wants to use, line number just stating which number line that you are editing and text highlighting that will enable code readibility. Design part of module will allow developer to drag-drop designing. Developer will just click on an element on any of palettes and dragging it to design form, it will work well. Adding a button or such an elemnt will change code part of module, as noted above thanks to the relation between two parts.

3.3 Project Module

Project module is a part of software that will help where to and how to save and organize files. It will depending on the system, will make default directory and subdirectories for setup. When opening a new project and or just opening a previously saved project will be held by project module part of the software.

3.4 Debugger Module

Debugger module has some powerful debugging capabilities that will help developer to detect his errors easily and recover them. It will use tect-highlighting as well to show there is something wrong with that part of code. Debugger module will also check code in such a wat that if an opening paranthesis is available and there is not corresponding closing paranthesis,i.e. there is a missing paranthesis, it will warn by changing the color of that code segment.

3.5 Database Module

Database module as in many other software projects is a little complicated and time consuming concept. However, it is maybe one of the most important concepts that should given much time to have an efficient database module which is working well and adoptable to other modules of the project.

One can not divide database module into parts, but in generally speaking and for some definition and clear understanding of the subject, we will try to tell it in such a way and give the important and inevitable parts in a reasonable manner.

Database module then can be explained in two parts consisting of user database and program software database. To give more specifics about user database, one can say that it provides some chance with user like connecting to his/her remote database and process some SQL job on it. There will be a connection form that will require a user name and password besides tunneling information or port number special to that server, and help user to connect his/her remote database. In addition, after connection is set to a database, a developer will easily implement some SQL codes via database tab in the GUI by selecting database name that s/he wants to work on.

To talk about the second part which is program software database, one can say that it is required as in many other software programs and will provide faster and dynamic implementations. It consist of much information on many sub patterns of the software from how to and where to save files, to how to store properties of the objects on design module. It also provide some easiness and dynamism for the design module of the project by storing all important code segments and line numbers in the code as of classes, methods and global functions.

4. USER INTERFACE DESIGN

User interface is such an important concept that it directly affects the user to decide to use the software or not, regardless of the quality of other parts of the software. No one wants to use a not-user-friendly software. So, we tried to design (and will go on trying) the graphical user interface (GUI) as much user-friendly as we could. Below, we will try to explain the GUI in detail and we will give some screenshots when needed.

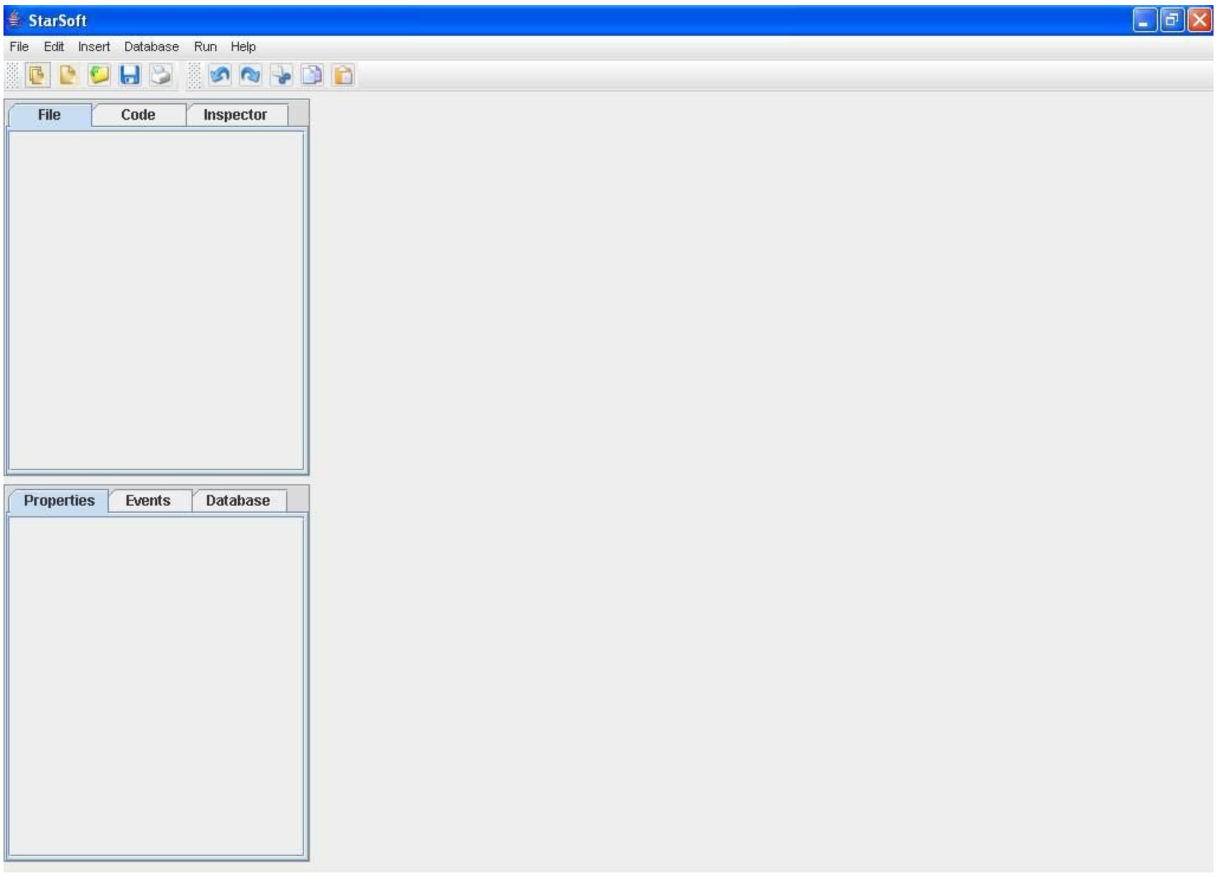

StarSoft
File Edit Insert Database Run Help
File Code Inspector
Properties Events Database

Figure 1 An empty program window

4.1 Menus

We will have 6 menus on the top of the application screen. These menus are “File”, “Edit”, “Insert”, “Database”, “Run”, and “Help”.

4.1.1 File Menu

In the “File” menu, we will have the usual menu items. These are “New Project”, “New File”, “Open”, “Save”, “Save as”, “Print”, “Recent Files”, and “Exit”.

“New Project” item will trigger a popup window, and in this window user will determine the project’s name and related initial inputs that must be necessary to create a new project.

“New File” item, like “New Project”, will trigger a popup window, and in this window user will determine the file’s name, type and other related initial inputs that is necessary to create a new file.

“Open” item, again, will trigger a popup window, and in this window user will be able to open either a project or a source file to edit.

“Save” item will save the file that is active if the file is saved before. Otherwise it will react as “Save as” item which is explained below.

“Save as” item will trigger a popup window, and in this window the location and name of the file to be saved will be determined.

“Print” item will popup a window, and this window will give the user the opportunity to handle print-related issues. This item is to get the hard copy of the source files.

“Recent Files” item will hold a list of recently edited files.

“Exit” item will close the program.

4.1.2 Edit Menu

Edit menu will have nine items which are: “Undo”, “Redo”, “Copy”, “Cut”, “Paste”, “Delete”, “Find”, “Find and Replace”, “Balance Braces”.

“Undo” will take back the last performed action.

“Redo” is reverse of “Undo”.

“Copy” copies the selected parts of the source code or components in the design menu to clipboard and does not change anything else.

“Cut” copies the selected parts of the source code or components in the design menu to clipboard and deletes the selected parts or components.

“Paste” puts the part or component in the clipboard to desired place.

“Delete” deletes the selected part or component.

“Find” popups a window to find a pattern in the source code of the active file.

“Find and Replace” popups a window to find a pattern in the source code of the active file and replaces these occurances with the specified pattern.

4.1.3 Insert Menu

“Insert” menu is a big menu. It has thirteen items and five of these items also has subitems. Even these subitems has some subitems too. To start somehow these thirteen items are “Tag”, “Image”, “Image Objects”, “Media”, “Table”, “Table Objects”, “Form”, “Hyperlink”, “Email Link”, “Named Anchor”, “Date”, “Comment”, and “HTML”.

“Tag” lists all the HTML tags which are classified according to their function in a popup window. Of course this is not only for listing. These tags can be used for entering these tags easier into the code.

“Image” popups a window to select the image to be inserted.

“Image Objects” has three subitems. These are “Image Placeholder”, “Rollover Image”, and “Navigation Bar”. “Image Placeholder”, as the name implies puts an image placeholder with the specified size and text. “Rollover Image” puts an image into the specified place and changes to another image when mouse is over. “Navigation Bar” creates a navigation bar with images. Image changes when the mouse is over or out. It can be useful in circumstances like one wants to show one photo of an album in every page.

“Media” has nine subitems. These are “Flash”, “Image Viewer”, “Flash Text”, “Flash Button”, “Flash Paper”, “Flash Video”, “Shockwave”, “Applet”, and “Active X”. These subitems are very self-explanatory that one can easily figure out what all of these is intended to do.

“Table” insert a table with the specified number of rows, columns and properties.

“Table Objects” has ten subitems. These are “Insert Row Above”, “Insert Row Below”, “Insert Column to the Left”, “Insert Column to the Right”, “Import Tabular Data”, “Table”, “TR”, “TH”, “TD”, and “Caption”. First four of these insert either rows or column. “Import Tabular Data” is supposed to parse delimited text files and put the data into the table. Last 5 subitems are actually inserts HTML tags in the source code.

“Form” has fourteen subitems. These are “Form”, “Text Field”, “Textarea”, “Button”, “Checkbox”, “Radio Button”, “List/Menu”, “File Field”, “Image Field”, “Hidden Field”, “Radio Group”, “Jump Menu”, “Field Set”, and “Label”. These subitems insert implied form objects.

“Hyperlink” inserts an hyperlink.

“Email Link” inserts an email link.

“Named Anchor” inserts a named anchor.

“Date” inserts the date in the specified format.

“Comment” gives the opportunity to insert comments.

“HTML” has six subitems. These are “Horizantal Rule”, “Frames”, “Text Objects”, “Script Objects”, “Head Tags”, and “Special Characters”. “Horizantal Rule” insert a horizantal rule. “Frames” has seventeen subitems. These are “Left”, “Right”, “Top”, “Bottom”, “Bottom Nested Left”, “Bottom Nested Right”, “Left Nested Top”, “Left Nested Bottom”, “Right Nested Bottom”, “Right Nested Top”, “Top and Bottom”, “Top Nested Left”, “Top Nested Right”, “Frameset”, “Frame”, “Floating Frame”, and “Noframes”. First thirteen of them insert a frame into the implied direction. Last four are tags to insert into source code. “Text Objects” has nineteen subitems. These are “Font”, “Bold”, “Italic”, “Strong”, “Em”, “Paragraph”, “Block Quote”, “Preformated Text”, “H1”, “H2”, “H3”, “Unordered List”, “Ordered List”, “List Item”, “Definition List”, “Definition Term”, “Definition”, “Abbreviation, and “Acronym”. These subitems are also self-explanatory. “Script Objects” has two subitems: “Script”, “NoScript”. These are for JavaScript. “Head Tags” has 6 subitems which are “Meta”, “Keywords”, “Description”, “Refresh”, “Base”, and

“Link”. “Special Characters” has thirteen subitems all for some special characters or symbols. These are “Line Break”, “Non-Breaking Space”, “Copyright”, “Registered”, “Trademark”, “Dollar”, “Pound”, “Yen”, Euro”, “Left Quote”, “Right Ouote”, “Em-Dash”, and “Other”. “Other” popups a window that has more special characters.

4.1.4 Database Menu

“Database” menu has 2 items: “Connect” and “Disconnect” for either connecting to a database or disconnecting from a connected database.

4.1.5 Run Menu

“Run menu” has 6 items: “Run Main Project”, “Debug Main Project”, “Run File”, “Debug File”, “Run History”, and “Debug History”.

“Run Main Project” runs the main project.

“Debug Main Project” debugs the main project.

“Run File” runs the active file.

“Debug File” debugs the active file.

“Run History” lists recently runned projects and files.

“Debug History” holds the list of recently debugged projects and files.

4.1.6 Help Menu

“Help menu” has 2 items: “Help Contents” and “About”.

“Help Contents” is for the documentation of the software.

“About” gives some information about the software, like the version.

4.2 Tabs

There six tabs in two seperate windows. These are “File”, “Code”, “Inspector”, “Properties”, “Events”, and “Database”.

4.2.1 File Tab

“File” tab gives an overview of the project and its folders and directories in a tree fashion.

4.2.2 Code Tab

"Code" tab gives an overview of the active file. Classes and global variables (including functions) are listed here in a tree fashion. The following image shows the code tab.

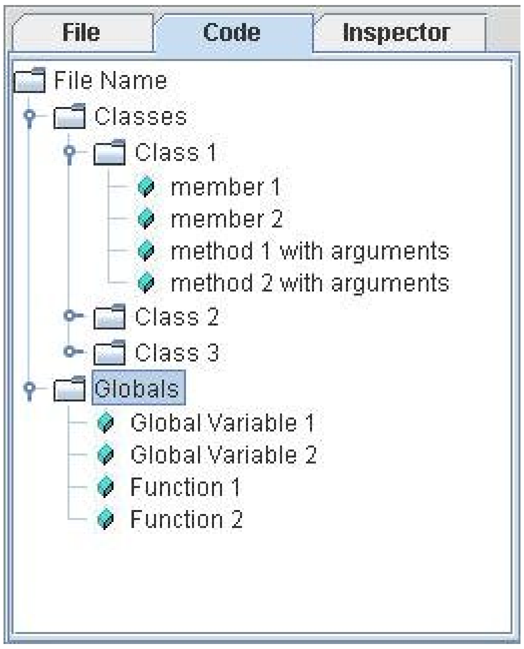


Figure 2 Code Tab

4.2.3 Inspector Tab

"Inspector" tab gives an overview of the objects or components like form objects in the active file in a tree fashion. The following image shows the inspector tab.

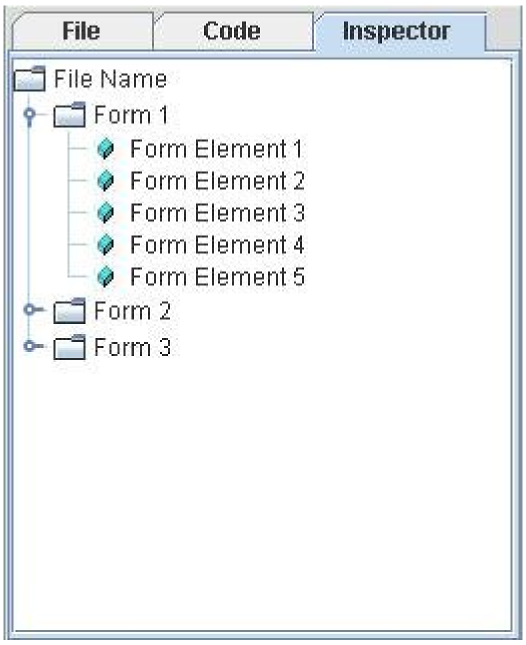
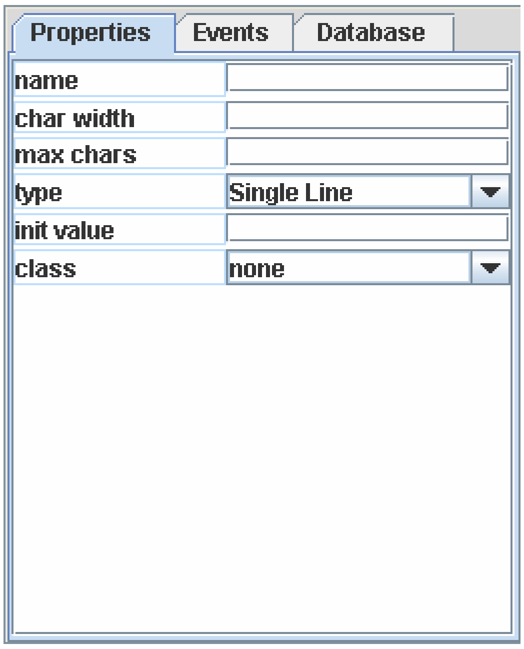


Figure 3 Inspector Tab

4.2.4 Properties Tab

"Properties" tab shows the properties of the active objects or components. The following image shows the properties tab.

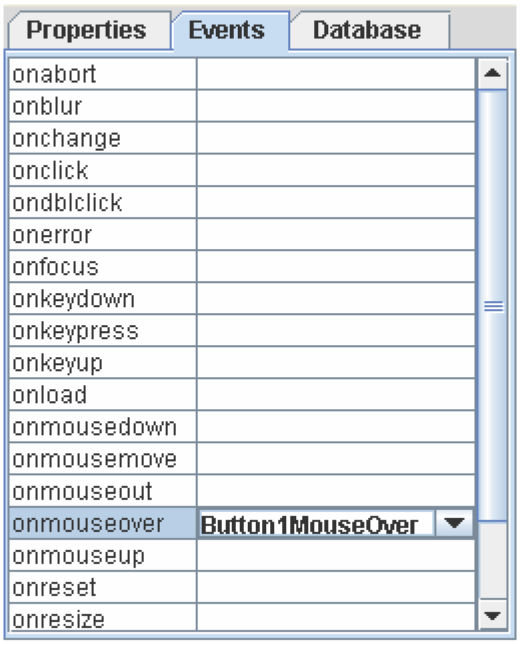


name	
char width	
max chars	
type	Single Line
init value	
class	none

Figure 4 Properties Tab

4.2.5 Events Tab

"Events" tab shows the functions that are related to actions of the active components or objects. The following image shows the events tab.

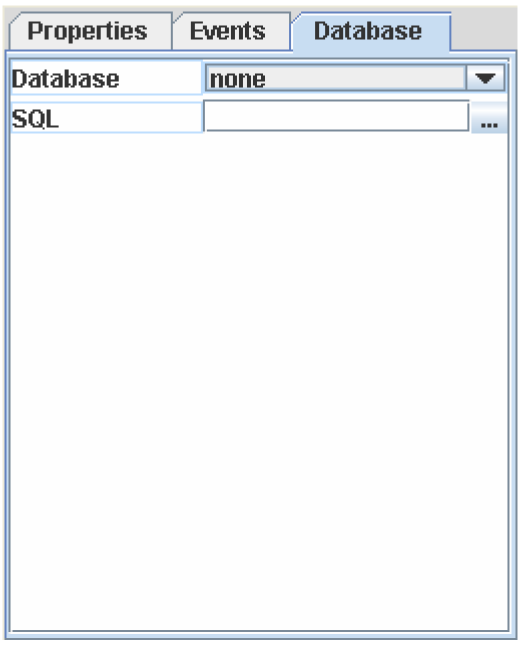


onabort	
onblur	
onchange	
onclick	
ondblclick	
onerror	
onfocus	
onkeydown	
onkeypress	
onkeyup	
onload	
onmousedown	
onmousemove	
onmouseout	
onmouseover	Button1MouseOver
onmouseup	
onreset	
onresize	

Figure 5 Events Tab

4.2.6 Database Tab

"Database" tab is to handle active objects' database related issues. The following image shows the database tab.

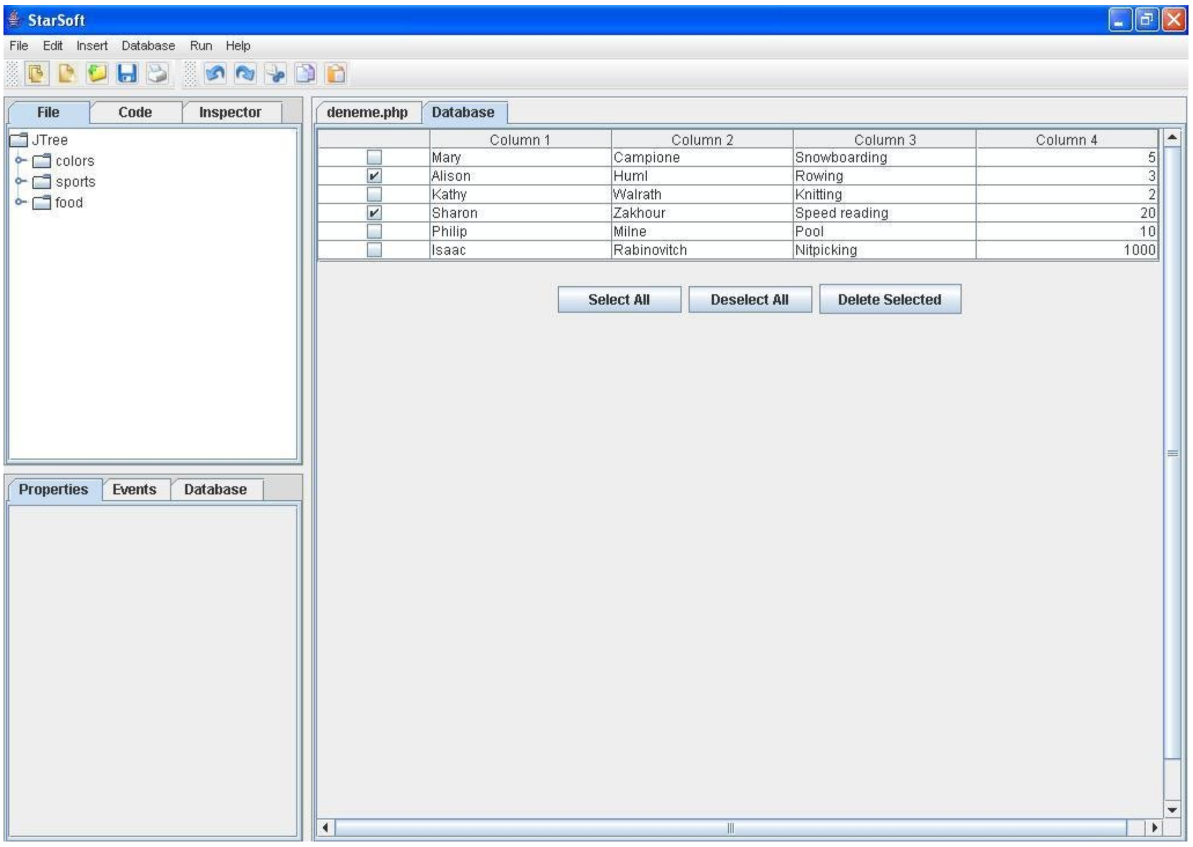


Database: none
SQL:

Figure 6 Database Tab

4.3 Main Panel

"Main Panel" will show all the opened files and database windows in a project in a tabbed fashion. The following image shows the main panel.


StarSoft
File Edit Insert Database Run Help
File Code Inspector
JTree
colors
sports
food
Properties Events Database
deneme.php Database
Select All Deselect All Delete Selected

	Column 1	Column 2	Column 3	Column 4
☐	Mary	Campione	Snowboarding	5
✓	Alison	Huml	Rowing	3
☐	Kathy	Walrath	Knitting	2
✓	Sharon	Zakhour	Speed reading	20
☐	Philip	Milne	Pool	10
☐	Isaac	Rabinovitch	Nitpicking	1000

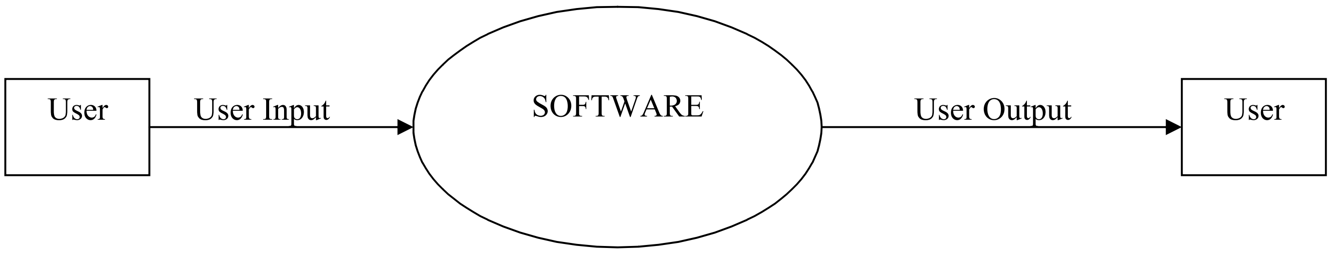
Figure 7 Main panel with 2 tabs

In figure 7 main panel has 2 tabs. First tab shows a file named "deneme.php". The second tab shows a database table. Database tables will be changable in this view. Editing a cell, deleting a row will be possible in here. Adding/deleting a column will be done in a seperate view because the database table in figure 7 can be a result of an SQL query as well as the whole table.

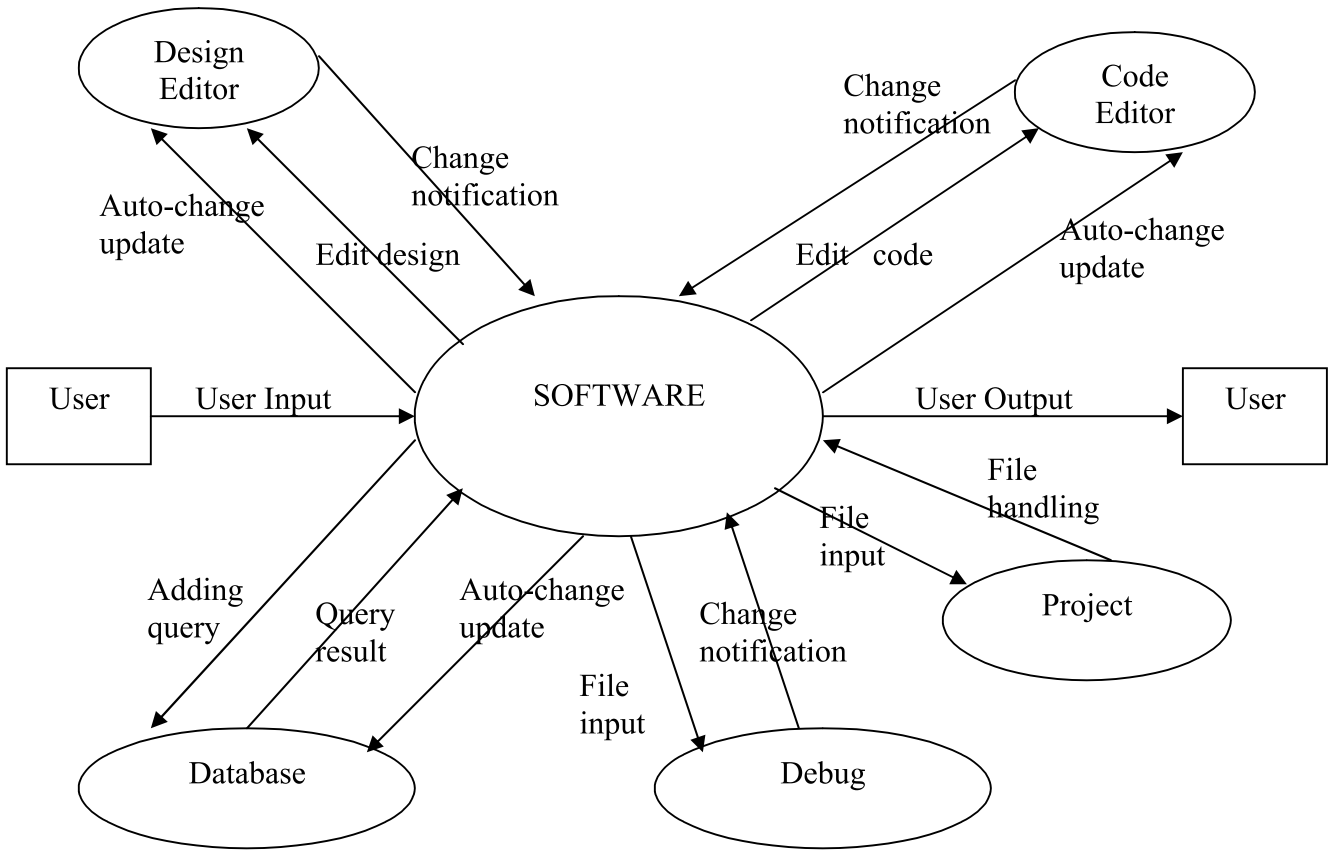
5. SYSTEM MODELING

5.1 DFD

5.1.1 DFD LEVEL0



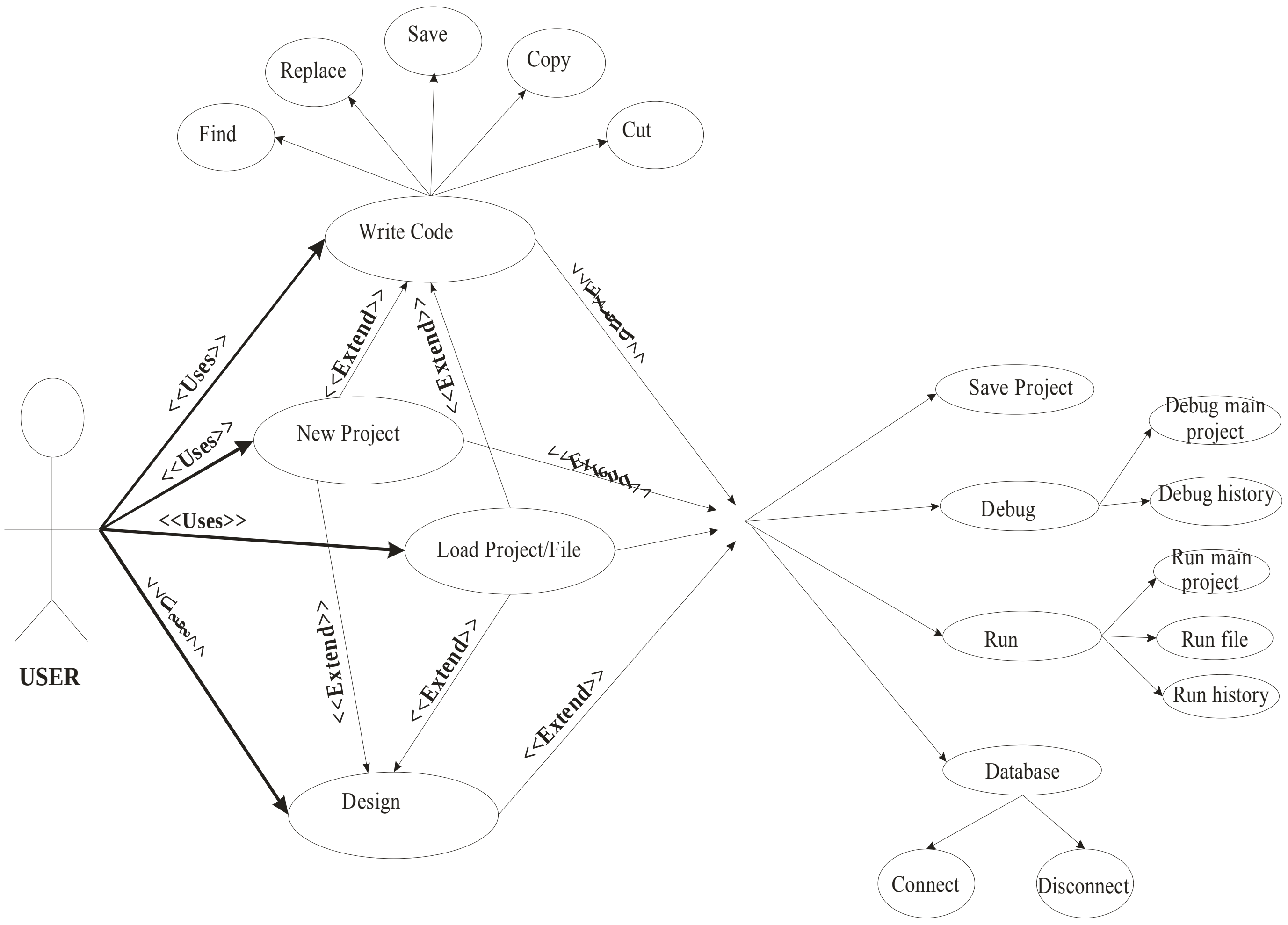
5.1.2 DFD LEVEL 1



Auto-change Update: A means of updates of change, and works when there is a change. For example there can be a change in code editor that is, developer may possibly add some codes to code editor, or in design editor that is, developer may possibly add some graphical elements to design environment. When there is such a situation, software automatically reflects new patterns to other parts and thus update.

Change Notification: This is a notification to software that there is a change in some parts and there must be some changes in other parts of studio, that is, reflections of previous changes must be applied to others. For example, if there is a change in or adding a graphical element to design editor, this must be reflected to code editor immediately.

5.2 Use-Case Diagram



New Project

Objective	To allow user to open a new project
PreCondition	--
Main Flow	1) The user interacts with the main window of the program . 2)User can open a new project by using selecting “New project” from menu bar under “File” menu

LOAD PROJECT/FILE

Objective	To allow user to load project/file previously saved
PreCondition	--
Main Flow	1) The user interacts with the main window of the program . 2)User loads a file by using selecting load from menu bar under File menu 3) After pressing “open file” or “open project” button standard browse window appears. User finds the desired project/file.

WRITE CODE

Objective	To allow user to modify code editorr
PreCondition	--
Main Flow	1) The users modify the code directly from here. . 2)User has som functionality like undo, redo, copy, cut, paste, find, find and replace.To use this functions user should press “Edit” button of menu bar select the function.

DESIGN

Objective	To allow user to modify design editor
PreCondition	--
Main Flow	1) The users modify the code design from here. .

SAVE PROJECT

Objective	To allow user to save current project
PreCondition	The project must be opened or a new project must opened.
Main Flow	1) The user interacts with the main window of the program. 2) User presses “save” butto n under “File” from menu bar. 3)After pressing “save” button a window appears and asks the name and location of the project.After selection and name user can save the Project

DATABASE

Objective	To allow user to connect or disconnect database
PreCondition	The project must be opened.
Main Flow	1) The user interacts with the main window of the program. 2) User presses “Database” from menu bar. 3) Under “Database”,user has two chocies:”Connect” is for the connect the database and “disconnect” is disconnect from this database.

RUN

Objective	To allow user to run the project,file or history.
PreCondition	The project must be opened or a new project must opened.
Main Flow	1) The user interacts with the main window of the program. 2) User presses “Run” from menu bar. 3) Under “Database”,user has three chocies to run:”Run main project” is for the current project, ”run file ” is for the selected file and “run history” is for the projects used recently.

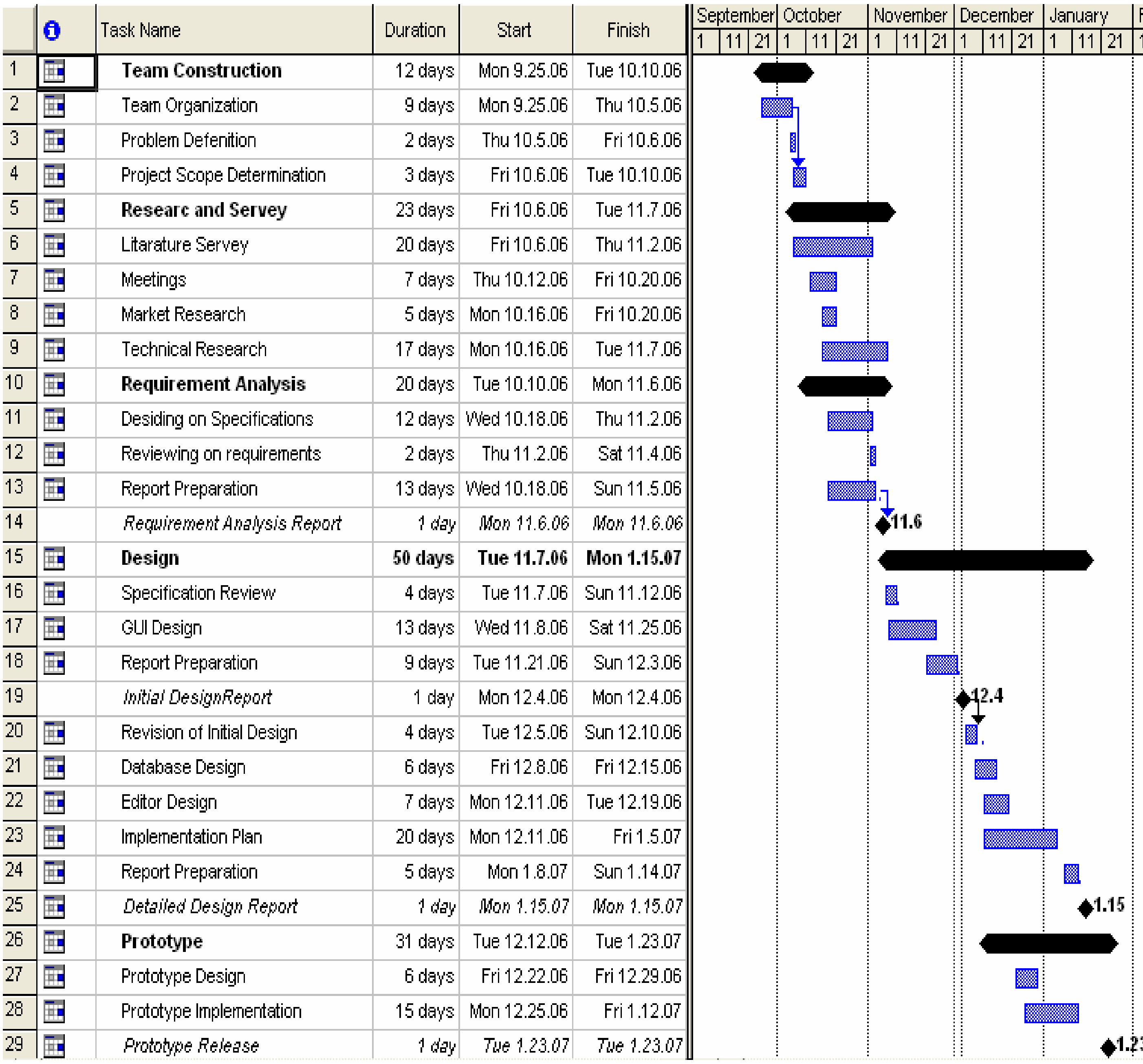
2.6 CLASS DIAGRAMS

```
class form_elements {
public:
        form_elements();
        void set_border_color(cons string color_value);
        void set_border_style(cons string style_type);
        void set_color(cons string color);
        void set_font_name(cons string font_name);
        void set_font_color(cons string font-color);
        void set_font_size(cons string font_size);
        void set_font_style(cons string font_style);

private:
        string border_color;
        string border_style;
        string color;
        string font_name;
        string font_color;
        string font_size;
        string font_style;
}
```

```
class form_process{
public:
        form_process();
        void save(void);
        void save_as(cons string new_path);
        void new_project(void);
        void open_project(void);
        string cut_code(string tmp_str);
        string copy_code(cons string tmp_str);
        void paste_code(string tmp_str);
        void undo_job(void);
        void redo_job(void);
        int find(string temp_char);
        void find_and_replace(string temp_char);
        void new_database(void);

private:
        string path="file:\\C://Program Files/Starsoft/"; //Path for some
        string temp; // For temporary process later
}
```



	ℹ	Task Name	Duration	Start	Finish
1	▦	**Team Construction**	12 days	Mon 9.25.06	Tue 10.10.06
2	▦	Team Organization	9 days	Mon 9.25.06	Thu 10.5.06
3	▦	Problem Defenition	2 days	Thu 10.5.06	Fri 10.6.06
4	▦	Project Scope Determination	3 days	Fri 10.6.06	Tue 10.10.06
5	▦	**Researc and Servey**	23 days	Fri 10.6.06	Tue 11.7.06
6	▦	Litarature Servey	20 days	Fri 10.6.06	Thu 11.2.06
7	▦	Meetings	7 days	Thu 10.12.06	Fri 10.20.06
8	▦	Market Research	5 days	Mon 10.16.06	Fri 10.20.06
9	▦	Technical Research	17 days	Mon 10.16.06	Tue 11.7.06
10	▦	**Requirement Analysis**	20 days	Tue 10.10.06	Mon 11.6.06
11	▦	Desiding on Specifications	12 days	Wed 10.18.06	Thu 11.2.06
12	▦	Reviewing on requirements	2 days	Thu 11.2.06	Sat 11.4.06
13	▦	Report Preparation	13 days	Wed 10.18.06	Sun 11.5.06
14		*Requirement Analysis Report*	*1 day*	*Mon 11.6.06*	*Mon 11.6.06*
15	▦	**Design**	**50 days**	**Tue 11.7.06**	**Mon 1.15.07**
16	▦	Specification Review	4 days	Tue 11.7.06	Sun 11.12.06
17	▦	GUI Design	13 days	Wed 11.8.06	Sat 11.25.06
18	▦	Report Preparation	9 days	Tue 11.21.06	Sun 12.3.06
19		*Initial DesignReport*	1 day	Mon 12.4.06	Mon 12.4.06
20	▦	Revision of Initial Design	4 days	Tue 12.5.06	Sun 12.10.06
21	▦	Database Design	6 days	Fri 12.8.06	Fri 12.15.06
22	▦	Editor Design	7 days	Mon 12.11.06	Tue 12.19.06
23	▦	Implementation Plan	20 days	Mon 12.11.06	Fri 1.5.07
24	▦	Report Preparation	5 days	Mon 1.8.07	Sun 1.14.07
25	▦	*Detailed Design Report*	*1 day*	*Mon 1.15.07*	*Mon 1.15.07*
26	▦	**Prototype**	31 days	Tue 12.12.06	Tue 1.23.07
27	▦	Prototype Design	6 days	Fri 12.22.06	Fri 12.29.06
28	▦	Prototype Implementation	15 days	Mon 12.25.06	Fri 1.12.07
29	▦	*Prototype Release*	*1 day*	*Tue 1.23.07*	*Tue 1.23.07*