CENG 490

STARSOFT

Final Design Report

By;
Mehmet ALBAYRAK
Ömer ESER
Özer GÜMÜŞ
Özge YAMASAN
Fatih YILDIRIM
# INDEX

1. INTRODUCTION .............................................................................................................................................................................................. 3
  1.1 PURPOSE OF THE DOCUMENT ...................................................................................................................................................................... 3
  1.2 SCOPE ................................................................................................................................................................................................... 3
  1.3 DESIGN CONSTRAINTS AND LIMITATIONS .................................................................................................................................................. 4
    1.3.1 Constraints .................................................................................................................................................................................................. 4
    1.3.2 Limitations .................................................................................................................................................................................................. 4
  1.4 GOALS AND OBJECTIVES .................................................................................................................................................................................. 4

2. DATA DESIGN ................................................................................................................................................................................................... 5
  2.1 DATA OBJECTS ................................................................................................................................................................................................... 5
  2.2 ER DIAGRAMS ................................................................................................................................................................................................... 10
  2.3 SEQUENCE DIAGRAMS .......................................................................................................................................................................................... 11
  2.4 DATA DICTIONARY ................................................................................................................................................................................................... 12
  2.5 INTERNAL SOFTWARE DATA STRUCTURE ...................................................................................................................................................... 18
  2.6 DATABASE DESCRIPTION ....................................................................................................................................................................................... 23
  2.7 DATABASE NORMALIZATION .................................................................................................................................................................................. 23

3. ARCHITECTURAL AND COMPONENT-LEVEL DESIGN ...................................................................................................................................................................................... 24
  3.1 STRUCTURE CHART ................................................................................................................................................................................................... 24
  3.2 EDITOR MODULE ................................................................................................................................................................................................... 24
  3.3 PROJECT MODULE................................................................................................................................................................................................... 25
  3.4 DEBUGGER MODULE ................................................................................................................................................................................................... 25
  3.5 DATABASE MODULE ................................................................................................................................................................................................... 25

4. SYSTEM MODELING ............................................................................................................................................................................................... 26
  4.1 DFD ....................................................................................................................................................................................................................... 26
    4.1.2 DFD LEVEL 1 ................................................................................................................................................................................................... 27
  4.2 USE CASE DIAGRAM ............................................................................................................................................................................................... 28
    4.2.1 New/Load Project ................................................................................................................................................................................................... 28
    4.2.2 Write Code ................................................................................................................................................................................................... 30
    4.2.3 Design .............................................................................................................................................................................................................. 31
    4.2.4 Debugging ................................................................................................................................................................................................... 32
    4.2.5 Run .............................................................................................................................................................................................................. 33

5. CLASS DIAGRAM ................................................................................................................................................................................................... 35
  5.1 MAIN_MANAGER CLASS ...................................................................................................................................................................................... 35
  5.2 FILE_OBJECT CLASS ......................................................................................................................................................................................... 36
  5.3 EDIT_OBJECT CLASS ....................................................................................................................................................................................... 36
  5.4 INSERT_OBJECT CLASS ................................................................................................................................................................................... 37
  5.5 DATABASE_OBJECT CLASS ............................................................................................................................................................................ 38
  5.6 RUN_OBJECT CLASS ....................................................................................................................................................................................... 39

6. USER INTERFACE DESIGN ...................................................................................................................................................................................... 41
  6.1 MENUS ....................................................................................................................................................................................................................... 42
    6.1.1 File Menu ................................................................................................................................................................................................... 42
    6.1.2 Edit Menu ................................................................................................................................................................................................... 43
    6.1.3 Insert Menu ................................................................................................................................................................................................... 44
    6.1.4 Database Menu ................................................................................................................................................................................................... 46
    6.1.5 Run Menu ................................................................................................................................................................................................... 46
    6.1.6 Help Menu ................................................................................................................................................................................................... 46
  6.2 TABS ....................................................................................................................................................................................................................... 47
    6.2.1 File Tab ................................................................................................................................................................................................... 47
    6.2.2 Code Tab ................................................................................................................................................................................................... 47
    6.2.3 Inspector Tab ................................................................................................................................................................................................... 48
    6.2.4 Properties Tab ................................................................................................................................................................................................... 49
    6.2.5 Events Tab ................................................................................................................................................................................................... 49
    6.2.6 Database Tab ................................................................................................................................................................................................... 50
  6.3 MAIN PANEL ................................................................................................................................................................................................... 50

7. GANTT CHART ................................................................................................................................................................................................... 51
1. INTRODUCTION

1.1 Purpose of the Document

This document is the Final 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 in more detail than the Initial Design Report. This document aims to establish a basis for the implementation phases. In the preparation of the document we found the inclusion of the following necessary:

- Modular Specifications
- UML Diagrams
- Screenshots
- Updated Schedule

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 necessary 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.
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 Form_Name together will be unique.

Form_Elements

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

- Form_Elements_ID
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_Moustdown
- On_Mousemove
- On_Mouseup
- On_Mouseover
- On_Mouseout
- On_Reset
- On_Resiz
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

Property_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 1 shows 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
2.3 Sequence Diagrams

User -> Main_Manager
File_Op()

Main_Manager -> File_Object
New_File()
Save
Exit()

File_Object
2.4 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.

<table>
<thead>
<tr>
<th><strong>File_ID</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>File_ID</td>
<td>Alias</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>It is assigned all files when they are created.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Each file has a unique File_ID.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>File_Name</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>File_Name</td>
<td>Alias</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>It is assigned all files when they are created.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Each file has a unique File_Name.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>INSPECTOR</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>INSPECTOR</td>
<td>Alias</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>Every file entity has inspector.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Inspector entity is investigating the File entity. Form names saved in it.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Inspector_ID</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Inspector_ID</td>
<td>Alias</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>Primary key of inspector.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Each inspector has a unique Inspector_ID.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Form_Name</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Form_Name</td>
<td>Alias</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>It is saved in INSPECTOR entity.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Name of the form.</td>
<td></td>
</tr>
</tbody>
</table>
**FORM ELEMENT**

<table>
<thead>
<tr>
<th>Name</th>
<th>FORM ELEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>When INSPECTOR takes action.</td>
</tr>
<tr>
<td>Description</td>
<td>Each form and their line number stored in it. It has also form elements name</td>
</tr>
</tbody>
</table>

**Form_Elements_ID**

<table>
<thead>
<tr>
<th>Name</th>
<th>Form_Elements_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>Primary key of FORM ELEMENT.</td>
</tr>
<tr>
<td>Description</td>
<td>Each FORM ELEMENT has a unique Form_Elements_ID</td>
</tr>
</tbody>
</table>

**Form_Elements_Name**

<table>
<thead>
<tr>
<th>Name</th>
<th>Form_Elements_Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>Member of FORM ELEMENTS</td>
</tr>
<tr>
<td>Description</td>
<td>It is string and stores the name of elements</td>
</tr>
</tbody>
</table>

**Line_Number**

<table>
<thead>
<tr>
<th>Name</th>
<th>Line_Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>-</td>
</tr>
<tr>
<td>Where / How used</td>
<td>Member of FORM ELEMENTS</td>
</tr>
<tr>
<td>Description</td>
<td>It is integer and showa the line number of each element.</td>
</tr>
</tbody>
</table>
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

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

CODE
Name CODE
Alias -
Where / How used Every file entity has CODE entity.
<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th><strong>It stores the code data of file.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code_ID</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Code_ID</td>
</tr>
<tr>
<td><strong>Alias</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Where / How used</strong></td>
<td>Primary key of CODE.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Each CODE has a unique Code_ID.</td>
</tr>
<tr>
<td><strong>Code_Name</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Code_Name</td>
</tr>
<tr>
<td><strong>Alias</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Where / How used</strong></td>
<td>Member of CODE entity</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>It is name of the classes or functions in code. It is unique</td>
</tr>
<tr>
<td><strong>Code_Type</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Code_Type</td>
</tr>
<tr>
<td><strong>Alias</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Where / How used</strong></td>
<td>Member of CODE entity</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>It is an integer: 1 or 2. 1 is used for classes 2 is used functions. It is unique</td>
</tr>
<tr>
<td><strong>Code_Line_Number</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Code_Line_Number</td>
</tr>
<tr>
<td><strong>Alias</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Where / How used</strong></td>
<td>Member of CODE entity</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>It is line number of code.</td>
</tr>
<tr>
<td><strong>MEMBERS</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>MEMBERS</td>
</tr>
<tr>
<td><strong>Alias</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Where / How used</strong></td>
<td>Every CODE entity has MEMBERS.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>It stores the variables and the functions of the each class. It has unique Code_ID</td>
</tr>
<tr>
<td><strong>Member_ID</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Member_ID</td>
</tr>
<tr>
<td><strong>Alias</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Where / How used</strong></td>
<td>Primary key of MEMBERS.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Each MEMBERS has a unique Member_ID.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Member_Name</td>
<td>Name of elements. It is unique.</td>
</tr>
<tr>
<td>Alias</td>
<td></td>
</tr>
<tr>
<td>Where / How used</td>
<td>Element of MEMBERS.</td>
</tr>
<tr>
<td>Database_ID</td>
<td>Each DATABASE has a unique Database_ID.</td>
</tr>
<tr>
<td>Name</td>
<td>Database_ID</td>
</tr>
<tr>
<td>Alias</td>
<td></td>
</tr>
<tr>
<td>Where / How used</td>
<td>Primary key of DATABASE.</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Member_Line_Number</td>
<td>It shows the line number of member.</td>
</tr>
<tr>
<td>Name</td>
<td>Member_Line_Number</td>
</tr>
<tr>
<td>Alias</td>
<td></td>
</tr>
<tr>
<td>Where / How used</td>
<td>Element of MEMBERS.</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>
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.

2.5 Internal Software Data Structure

Project

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

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

<table>
<thead>
<tr>
<th>Data</th>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>File_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>File_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>Project_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
</tbody>
</table>

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),
);

<table>
<thead>
<tr>
<th>Data</th>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspector_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Form_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>File_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
</tbody>
</table>
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),
);

<table>
<thead>
<tr>
<th>Data</th>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form_Elements_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Form_Elements_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>Line_Number</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Inspector_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
</tbody>
</table>

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_Mousedown INTEGER,
On_Mousemove INTEGER,
On_Mouseup INTEGER,
On_Mouseover INTEGER,
On_Mouseout INTEGER,
On_Reset INTEGER,
On_Resize INTEGER,
);
<table>
<thead>
<tr>
<th>Data</th>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Property_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>Border_Color</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Border_Style</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Color</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Font_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>Font_COLOR</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Font_Size</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Font_STYLE</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
</tbody>
</table>

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
<table>
<thead>
<tr>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties_ID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Property_Name</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Border_Color</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Border_Style</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Color</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Font_Name</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Font_COLOR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Font_Size</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Font_STYLE</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>
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),
);

<table>
<thead>
<tr>
<th>Data</th>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Code_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>Code_Line_Number</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Code_Type</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>File_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
</tbody>
</table>

Members

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

<table>
<thead>
<tr>
<th>Data</th>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Member_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>Member_Line_Number</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Code_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
</tbody>
</table>
Database

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

<table>
<thead>
<tr>
<th>Data</th>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Database_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>Project_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Data</th>
<th>Type&amp;Size</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
<tr>
<td>Table_Name</td>
<td>VARCHAR</td>
<td>Text</td>
</tr>
<tr>
<td>Database_ID</td>
<td>INTEGER</td>
<td>Number</td>
</tr>
</tbody>
</table>

### 2.6 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.

### 2.7 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
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 component-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 readability. 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 element 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. SYSTEM MODELING

4.1 DFD

4.1.1 DFD LEVEL0

![DFD Diagram]

```mermaid
graph TD
User --> User Input --> SOFTWARE
SOFTWARE --> User Output --> Software
Software --> User
```
4.1.2 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.

4.2 Use Case Diagram

4.2.1 New/Load Project
### New Project

<table>
<thead>
<tr>
<th>Objective</th>
<th>To allow user to open a new project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PreCondition</strong></td>
<td>--</td>
</tr>
</tbody>
</table>
| **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

<table>
<thead>
<tr>
<th>Objective</th>
<th>To allow user to load project/file previously saved</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PreCondition</strong></td>
<td>--</td>
</tr>
</tbody>
</table>
| **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. |
4.2.2 Write Code

USER

Write Code

Save Project

<<Extend>>

Debug

<<Extend>>

Run

<<Extend>>

Database

USER

Write Code

Copy

Find

Find and Replace

Cut

Save
**Objective**
- To allow user to modify code editor

**PreCondition**
- 

**Main Flow**
1) The users modify the code directly from here.
2) User has some functionality like undo, redo, copy, cut, paste, find, find and replace. To use these functions user should press “Edit” button of menu bar select the function.

---

**SAVE PROJECT**

<table>
<thead>
<tr>
<th>Objective</th>
<th>To allow user to save current project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PreCondition</strong></td>
<td>The project must be opened or a new project must opened.</td>
</tr>
<tr>
<td><strong>Main Flow</strong></td>
<td>1) The user interacts with the main window of the program.</td>
</tr>
<tr>
<td></td>
<td>2) User presses “save” button under “File” from menu bar.</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
</tbody>
</table>

---

4.2.3 Design
### DESIGN

<table>
<thead>
<tr>
<th>Objective</th>
<th>To allow user to modify design editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>PreCondition</td>
<td>--</td>
</tr>
<tr>
<td>Main Flow</td>
<td>1) The users modify the code design from here.</td>
</tr>
</tbody>
</table>

#### 4.2.4 Debugging
### DATABASE

<table>
<thead>
<tr>
<th><strong>Objective</strong></th>
<th>To allow user to connect or disconnect database</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PreCondition</strong></td>
<td>The project must be opened.</td>
</tr>
</tbody>
</table>
| **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 choices: “Connect” is for the connect the database and “disconnect” is disconnect from this database. |

#### 4.2.5 Run

```
USER
```

```
Run

Run Main Project
Run File
Run History
```

<<Extend>>
<table>
<thead>
<tr>
<th><strong>Objective</strong></th>
<th>To allow user to run the project, file or history.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PreCondition</strong></td>
<td>The project must be opened or a new project must opened.</td>
</tr>
</tbody>
</table>
| **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 choices 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. |
5. CLASS DIAGRAM

<< Implementation Class >>
DATABASE_OBJECT

- User Name
- Password
- Table_Id
- table_str
- Def_Db_Type
- Curr_Status

- Database.Op()
- Connect()
- Disconnect()
- Set_Db_Type()
- Select_Db()
- Set_Curr_Status()
- Set_Table() (optional)
- Create Table()
- Update_Db()
- Edit_Tables()

Calls

<< Implementation Class >>
MAIN_MANAGER

- Project_ID
- Project_Name
- Directory_Default
- CP_Type

- MAIN_MANAGER()
- File.Op()
- Edit.OP()
- Insert.OP()

Calls

<< Implementation Class >>
EDIT_OBJECT

- Obj_ID
- Temp.Str
- Default.Str

- Edit.Op()
- Undo_Process()
- Redo_Process()
- Cut()
- Copy()
- Paste()
- Delete()
- Select()
- Find()
- Find.Replace()
- Balance.Braces()

Calls

<< Implementation Class >>
INSERT_OBJECT

- Obj_ID
- Obj.Type
- Obj_Name
- Temp.Str

- Insert.OP()
- Cut.Obj.Type()
- Ins.Proof()
- Ins.Tag()
- Ins.Imge()
- Ins.Imge.Obj()
- Ins.Media()
- Ins.Table()
- Ins.Tb.Object()
- Ins.Form()
- Ins.Hyperlink()
- Ins.Anchor()
- Ins.Comment()
- Ins.Html()
Class diagram above is mainly show the relation of modules and operations which are completely explained in architectural and component-level design part of the report. There is a main manager class that is certainly calling some other classes and thus is is managing all that above in someway.

5.1. MAIN_MANAGER CLASS

This class has some elements that are private to this class. These elements are:

- **Project_Id**: This is a integer variable that will contain the id of the currently working project.
- **Project_Name**: This is a string variable keeping the name of currently working project.
- **Directory_Default**: This is a string variable that hold the default directory that will be used for saving and further processing of the project.
- **Op_Type**: It is a string variable to hold what operation will be done.

And also methods belonging to this class are:

- **Main_Manager()**: This is default constructor method of Main_Manager class.
- **File_Op()**: This method will call another class file object class
- **Edit_Op()**: This method will call another class edit object class
- **Insert_Op()**: This method will call another class insert object class
- **Database_Op()**: This method will call another class database object class
- **Run_Debug_Op()**: This method will call another class run object class
- **Get.Op_Type()**: This method will get the operation type.

5.2. FILE_OBJECT CLASS
This class has some elements that are private to this class. These elements are:

---

**File_Id:** This is an integer variable that will hold id the of the file that some operation will be done.

**Project_Id:** This is an integer variable that will hold id the of the project that some operation will be done.

**File_Name:** This is an string variable that will hold name the of the file that some operation will be done.

And also methods belonging to this class are:

- **File_Op:** This is default constructor method of file_object class.
- **New_File:** This is a method to create a new file.
- **New_Project:** This is a method to create a new project.
- **Open_File:** This is a method to open a currently existing file.
- **Open_Project:** This is a method to open currently existing project.
- **Open_Recent:** This is a method to open recently used project or file.
- **Save:** This is a method to save files or projects.
- **Save_As:** This is a method to save files or projects in a selected directory.
- **Print:** This is a method to print the currently working document.
- **Exit:** This is a method to exist the file and project.

---

### 5.3. EDIT_OBJECT CLASS

This class has some elements that are private to this class. These elements are:

---

**Obj_Id:** This is an integer variable that will hold id the of the object that some changes will be done.
Temp_Str: This is a string variable that will hold some text which will be used for cut, copy and for some other methods of the used.

Default_Str: This is a string variable that will hold some text and like Temp_Str, will be used in some methods like Find_Replace()

And also methods belonging to this class are:

Edit_Op(): This is default constructor method of edit_object class.
Undo_Process(): This is a method to undo jobs done lastly.
Redo_Process(): This is a method to redo jobs done lastly.
Cut(): This is a method to cut a selected text or object.
Copy(): This is a method to copy a selected text or object.
Paste(): This is a method to paste a copied/cutted text or object
Delete(): This is a method to delete a selected text or object.
Select(): This is a method to select all text or objects.
Find(): This is a method to find some text.
Find_Replac: :e() This is a method to find some text and replace will some other text given by developer.
Balance_Braces(): This is a method to balance braces.

5.4. INSERT_OBJECT CLASS

This class has some elements that are private to this class. These elements are:

Obj_Id: This is an integer variable that will hold id the of the object that will be inserted.

Obj_Type: This is a string variable that will hold type of the object that will be inserted.
Obj_Name: This is a string variable that will hold name of the object that will be inserted.

Temp_Str: This is a string variable that will be used in some methods for manipulation of data and etc.

And also methods belonging to this class are:

Insert_Op(): This is default constructor method of insert_object class.
Get_Obj_Type(): This is a method to get the type of the object that will be inserted.
Ins_PreDef(): This is a method to insert some predefined script.
Ins_Tag(): This is a method to insert tags.
Ins_Image(): This is a method to insert image.
Ins_Image_Obj(): This is a method to insert some image objects.
Ins_Media(): This is a method to insert media.
Ins_Table(): This is a method to insert a table.
Ins_Tb_Object(): This is a method to insert a table object.
Ins_Form(): This is a method to insert a form and form objects.
Ins_Hyperlink(): This is a method to insert a hyperlink.
Ins_Anchor(): This is a method to insert a name anchor.
Ins_Comment(): This is a method to insert some comments.
Ins_Html(): This is a method to insert some HTML codes.

5.5. DATABASE_OBJECT CLASS

This class has some elements that are private to this class. These elements are:

User_Name: This is a string variable that will hold username to connect to a database.
Passwd: This is a string variable that will hold password to connect to a database.

Table_Id: This is an integer variable that will hold the id of the table which will be created or processed.

Table_Str: This is a string variable that will hold text for table processing.

Def_Db_Type: This is a string variable that will hold the database type that user will connect.

Conn_Status: This is a string variable that will hold the connection status to a database.

And also methods belonging to this class are:

Database_Op(): This is default constructor method of database_object class.
Connect(): This is a method to connect to a database.
Disconnect(): This is a method to disconnect from a database.
Get_Db_Type(): This is a method to get database type of the database that will be connected.
Select_Db(): This is a method to select database name from connected database.
Get_Conn_Status(): This is a method to get the connection status to a database.
Get_Tables(): This is a method to get the names of tables from a database.
Create_Table(): This is a method to create a new database table.
Update_Db(): This is a method to update the changes that will be done.
Edit_Tables(): This is a method to edit and make some changes to database tables.

5.6. RUN_OBJECT CLASS

This class has some elements that are private to this class. These elements are:

Process_Type: This is a string variable that will hold the type of process which is either a run or debug process.
Temp_Str: This is a string variable that will be used in debugging process.

Break_Point: This is an integer array that will hold line numbers which are with break points.

Highlight_Str: This is a string variable that will hold highlighted text lines.

Out_Str: This is a string variable that will hold output of some debugging process.

And also methods belonging to this class are:

Run_Debug_Op(): This is default constructor method of run_object class.
Run(): This is a method to run the project or file.
Debug(): This is a method to debug the project or file.
Get_Pr_Type(): This is a method to get process type whether to run or debug the project or file.
Get_Run_Hist(): This is a method to get the results of previous runs of the project or file.
Get_Debug_Hist(): This is a method to get the results of previous debugs of the project or file.
Put_Break_Pnts(): This is a method to put breakpoints determined by developer during debug process.
Highlight(): This is a method to highlight some special texts or break point lines during debugging.

6. 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.
6.1 Menus

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

6.1.1 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.

6.1.2 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 occurrances with the specified pattern.
6.1.3 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.

“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.

“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 Quote”, “Em-Dash”, and “Other”. “Other” popups a window that has more special characters.

6.1.4 Database Menu

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

6.1.5 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.

6.1.6 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.
There are six tabs in two separate windows. These are “File”, “Code”, “Inspector”, “Properties”, “Events”, and “Database”.

### 6.2.1 File Tab

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

### 6.2.2 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.
6.2.3 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.
6.2.4 Properties Tab

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

![Figure 4 Properties Tab](image)

6.2.5 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.

![Figure 5 Events Tab](image)
6.2.6 Database Tab

“Database” tab is to handle active objects’ database related issues. The following image shows the database tab.

![Database Tab Image](image-url)

Figure 6 Database Tab

6.3 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.

![Main Panel Image](image-url)
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 separate view because the database table in figure 7 can be a result of an SQL query as well as the whole table.

7. GANNT CHART
<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team Construction</td>
<td>12 days</td>
<td>Mon 9.25.06</td>
<td>Tue 10.10.06</td>
</tr>
<tr>
<td>Team Organization</td>
<td>9 days</td>
<td>Mon 9.25.06</td>
<td>Thu 10.5.06</td>
</tr>
<tr>
<td>Problem Definition</td>
<td>2 days</td>
<td>Thu 10.5.06</td>
<td>Fri 10.6.06</td>
</tr>
<tr>
<td>Project Scope Determination</td>
<td>3 days</td>
<td>Fri 10.6.06</td>
<td>Tue 10.10.06</td>
</tr>
<tr>
<td>Research and Survey</td>
<td>24 days</td>
<td>Fri 10.6.06</td>
<td>Tue 11.7.06</td>
</tr>
<tr>
<td>Literature Survey</td>
<td>20 days</td>
<td>Fri 10.6.06</td>
<td>Thu 11.2.06</td>
</tr>
<tr>
<td>Meetings</td>
<td>7 days</td>
<td>Thu 10.12.06</td>
<td>Fri 10.20.06</td>
</tr>
<tr>
<td>Market Research</td>
<td>5 days</td>
<td>Mon 10.16.06</td>
<td>Fri 10.20.06</td>
</tr>
<tr>
<td>Technical Research</td>
<td>18 days</td>
<td>Mon 10.16.06</td>
<td>Tue 11.7.06</td>
</tr>
<tr>
<td>Requirement Analysis</td>
<td>21 days</td>
<td>Tue 10.10.06</td>
<td>Mon 11.6.06</td>
</tr>
<tr>
<td>Desiding on Specification</td>
<td>12 days</td>
<td>Wed 10.18.06</td>
<td>Thu 11.2.06</td>
</tr>
<tr>
<td>Reviewing on Requirements</td>
<td>2 days</td>
<td>Thu 11.2.06</td>
<td>Sat 11.4.06</td>
</tr>
<tr>
<td>Report Preparation</td>
<td>12 days</td>
<td>Wed 10.18.06</td>
<td>Thu 11.2.06</td>
</tr>
<tr>
<td>Requirement Analysis Report</td>
<td>1 day</td>
<td>Mon 11.6.06</td>
<td>Mon 11.6.06</td>
</tr>
<tr>
<td>Design</td>
<td>50 days</td>
<td>Tue 11.7.06</td>
<td>Mon 11.15.07</td>
</tr>
<tr>
<td>Specification Review</td>
<td>4 days</td>
<td>Tue 11.7.06</td>
<td>Sun 11.12.06</td>
</tr>
<tr>
<td>GUI Design</td>
<td>13 days</td>
<td>Wed 11.8.06</td>
<td>Sat 11.25.06</td>
</tr>
<tr>
<td>Report Preparation</td>
<td>9 days</td>
<td>Tue 11.21.06</td>
<td>Sun 12.3.06</td>
</tr>
<tr>
<td>Database Design</td>
<td>6 days</td>
<td>Thu 11.23.06</td>
<td>Thu 11.30.06</td>
</tr>
<tr>
<td>Initial Design Report</td>
<td>1 day</td>
<td>Mon 12.4.06</td>
<td>Mon 12.4.06</td>
</tr>
<tr>
<td>Revision of Initial Design Report</td>
<td>4 days</td>
<td>Tue 12.5.06</td>
<td>Sun 12.10.06</td>
</tr>
<tr>
<td>Database GUI Design</td>
<td>23 days</td>
<td>Tue 11.21.06</td>
<td>Thu 12.21.06</td>
</tr>
<tr>
<td>Editor Design</td>
<td>7 days</td>
<td>Mon 12.11.06</td>
<td>Tue 12.19.06</td>
</tr>
<tr>
<td>Task Name</td>
<td>Duration</td>
<td>Start</td>
<td>Finish</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Editor Design</td>
<td>7 days</td>
<td>Mon 12.11.06</td>
<td>Tue 12.19.06</td>
</tr>
<tr>
<td>Implantation Plan</td>
<td>20 days</td>
<td>Mon 12.11.06</td>
<td>Fri 1.5.07</td>
</tr>
<tr>
<td>Report Preparation</td>
<td>11 days</td>
<td>Wed 1.3.07</td>
<td>Wed 1.17.07</td>
</tr>
<tr>
<td>Detailed Design Report</td>
<td>1 day</td>
<td>Thu 1.18.07</td>
<td>Thu 1.18.07</td>
</tr>
<tr>
<td>Adding Functionality</td>
<td>38 days</td>
<td>Wed 11.29.06</td>
<td>Sun 1.21.07</td>
</tr>
<tr>
<td>Open New File</td>
<td>3 days</td>
<td>Wed 11.29.06</td>
<td>Fri 12.1.06</td>
</tr>
<tr>
<td>Open New Project</td>
<td>7 days</td>
<td>Mon 12.4.06</td>
<td>Tue 12.12.06</td>
</tr>
<tr>
<td>Save File</td>
<td>7 days</td>
<td>Mon 12.11.06</td>
<td>Tue 12.19.06</td>
</tr>
<tr>
<td>Save Project</td>
<td>5 days</td>
<td>Fri 12.15.06</td>
<td>Thu 12.21.06</td>
</tr>
<tr>
<td>Close Tab</td>
<td>5 days</td>
<td>Wed 12.20.06</td>
<td>Tue 12.26.06</td>
</tr>
<tr>
<td>Undo Redo</td>
<td>4 days</td>
<td>Fri 12.22.06</td>
<td>Wed 12.27.06</td>
</tr>
<tr>
<td>Copy Paste Cut</td>
<td>3 days</td>
<td>Fri 12.29.06</td>
<td>Tue 1.2.07</td>
</tr>
<tr>
<td>Short Cuts</td>
<td>4 days</td>
<td>Thu 1.4.07</td>
<td>Tue 1.9.07</td>
</tr>
<tr>
<td>Save All</td>
<td>4 days</td>
<td>Tue 1.9.07</td>
<td>Sun 1.14.07</td>
</tr>
<tr>
<td>Database Connection</td>
<td>10 days</td>
<td>Mon 1.8.07</td>
<td>Sun 1.21.07</td>
</tr>
<tr>
<td>Prototype</td>
<td>31 days</td>
<td>Tue 12.12.06</td>
<td>Tue 1.23.07</td>
</tr>
<tr>
<td>Prototype Design</td>
<td>6 days</td>
<td>Fri 12.22.06</td>
<td>Fri 12.29.06</td>
</tr>
<tr>
<td>Prototype Implementation</td>
<td>1 day</td>
<td>Fri 1.12.07</td>
<td>Fri 1.12.07</td>
</tr>
<tr>
<td>Prototype Release</td>
<td>1 day</td>
<td>Tue 1.23.07</td>
<td>Tue 1.23.07</td>
</tr>
</tbody>
</table>