METU

Computer Engineering

AKAMAI WEB TOOLKIT

FINAL DESIGN REPORT

Members of the Team:

•	Ahmet Emin Tosun	→	e141801@metu.edu.tr
•	Uğur Can Tekin	→	e134800@metu.edu.tr
•	Hasan İşler	→	e134758@metu.edu.tr
•	Vedat Şengül	→	e129829@metu.edu.tr
•	Muhammet Yavuz Aşık	→	e134713@metu.edu.tr

TABLE OF CONTENTS:

1 INTRODUCTION

1.1 Purpose and Scope of the Document

The purpose of this document is to state the constraints of the design specifications of the project. We have built this design document on to the initial design document we had prepared before. Note that even though we have used our initial design as a basis, we have revised it thoroughly in light of the advices and criticism received from our guides. It is an extension to the initial design specifications of the project for the designing phase by covering the further design approaches and analyzes our design processes.

In this report, we intend to give detailed information about how our solutions fulfill the problem requirements. We will present the revised content of our initial design report with some additional explanation, more detailed module descriptions and subtasks, our development plan and updated schedule. During our studies on this report, we have developed our sight to the problem and to the solution. This document contains detailed design description of the Akamai Web Toolkit. This final design of the project focuses on more detailed design considerations than the initial design and fullils the missing parts of the previous one. Therefore, beginning from the highest abstraction level of the system, this phase decreases the abstraction by covering the engineering details of each level until there is nothing to be left unclear for implementing a practical solution for the problem.

We will present our project’s modular specification and UML diagrams (use-case, class, data flow diagram, data dictionary and sequence diagrams) through which our understanding of the system improves.

In the project we want to serve the Ajax users a user friendly Ajax development environment. We will develop a system that will collect beneficial parts of current Ajax IDEs on the market and other similar developing studios. In this developing tool the scope is providing the web tool‘s users a graphical development environment for web pages with Ajax and providing them a usable interface for doing their works easily and in an understandable way. Our development tool will be able to recognize JavaScript CSS and HTML syntaxes. In our web developing tool the opportunity of writing new scripts, editing and deleting them is a crucial part; so we have to apply them in our project. While working with database, the database connection must be supplied. They can connect the database, send, and receive the data from their own databases. On the other hand implementing server-side actions in order to have the property of executing the writing scripts is an important part of our project scope. For making easier the interface of the working condition, the drag and drop objects, ready components must be served to the users and also at the same time it will support highlighting keywords of the related language. This feature of our program will help and make easy to see the different parts of the source codes writing by web developers. For checking the errors of the source that have been written must be checked by this tool and must underline the problematic parts. The error check tool must be warning to the user about the wrong parts of his codes and give a chance to see and correct them. Furthermore we must supply the user's code editing with the editing option.

1.2 Project Definition

AJAX stands for Asynchronous JavaScript and XML and is a web development technique for creating interactive web applications. Ajax's most appealing characteristic is its asynchronous nature, which means it can do all of this without having to refresh the page. This allows you to update portions of a page based upon user events. With AJAX, we only get the data from the server that we absolutely need, not the whole page. More importantly, data can be posted to and retrieved from the server after the entire page is loaded. This can be leveraged in creative and powerful ways to create a more fluid browsing experience. It can send as well as receive information in a variety of formats, including XML, HTML, and even text files. The technologies that are used to build AJAX web applications encompass a number of different programming domains, so AJAX development is neither as straightforward as regular applications development, nor as easy as old web development.

1.3 Design Constraints

1.3.1 Project Schedule

Project scheduling is one of the most important subtasks of project management. We have to finish the project until beginning of June 2007 so this is one of the main major constraints for our project. We have a lot of phase firstly we have designing, secondly building and finally test the project during this period. In this report, there is a Gannt Chart for the rest of the project.

Our Gannt chart can be seen under Appendix-A.

1.3.2 Language constraints

We are going to implement our project in Java Eclipse. This choice helps us while designing interface of our project. Also most of the open source libraries are written in Java. We thought that we can find more sources with this language as well.

1.3.3 User Interface

User friendly menus will be created. Easy to use and simple menus will be taken into consideration. We will work on interface for making easy to access to all the components. We believe that our program will be one step ahead from similar programs with this feature.

1.4 Usage Areas

Companies that works on web development area: Ajax is a very popular web development toolkit. So the companies that works on web developing can use our tool kit. Before the design process, we done market research and learn these related companies needs and expectations from this kind of tool. So we want to serve a usable tool kit to the market.

Students: The students who is interested in web development and web development in Ajax can choose our tool kit for their webpage. So we want to serve this tool kit from beginner level to the expert level. So students can use our program both for learning and self improvement as well. Today in the web design market, students are choosen for the companies for part time or full time jobs in the web development area. So we cannot overlook students in this case. For this reason we do market research on the computer science students and learn their needs before prepared our design in Akamai Web Toolkit.

Web-Designers: Web designers who work individually in the market for web development area are increasing today. So we cannot overlook them. They also uses different kinds of tool kit today. In the market research we asked for their needs and we learn them. In the market today companies gives the work of web development of their own web page to this kind of people and want to reduce the expense of this. To be chosed by this kind of people will be a good job for us. By this way we can improve our kit's popularity.

1.5 Design Goals

Our design goal is to achieve a well structured system that is functionally, logically and physically cohesive. This type of system is helpful for us to visualize, specify, construct and document it from di_erent, yet interrelated, aspects. Also, unavoidable changes can be tolerated easily when working on this kind of system.

In order to create a high performance and useful development tool the following goals will be the basis of our developing manner:

- **Performance:**

In Akamai Web Toolkit opens an .exe in a couple seconds. It doesn't require too much hardware requirement therefore it is fast and easy to develop a webpage in AWT. AWT will run and compile fast enough to meet the developer's needs.

• **Well organized code**:

By the help of using Java, an object oriented approach; we are planning to create a well organized code. As a result of this, our tool will be easily maintainable and extendible. We choose component based programming as a process model. Therefore, new features can be added easily. Our user can connect his database by the connect function and get data from the database, send and modify it.

• **Usability:**   Creating an easy to use development is the major of our main aims, because of the fact that the end users have limited knowledge of web developing techniques. Our end users include students and who starting a new web developing websites and also for companies that works on web development area. AWT will be useful program for all type of web developers and our program will support all browsers. (Firefox, Internet Explorer, Safari, Opera…)

• **Reliability**

The tool will be tested in every part of the implementation phase, thus crashes will be tried to be avoided.

• **Satisfying user needs:**

Most important area that our tool will be used is the web developers. Thus we have to consider the needs of web developers while he/she is using the development.   AWT's interface will be a user friendly one, so users can easily find what they need in the program.

1.6 Achieved Final Design Goals:

There were critical goals which should be achieved by the final design report. They were;

1. Data Flow diagrams, State Transition Diagrams and Graphical User Interface part of the system revised again and again in the final design of our project. Their consistency checked against newly added parts to the overall architecture.

2. In the initial design all Data Flow Diagrams and Sequence Diagram drawn carefully and they were revised to correct the errors in it. Also for each diagram all the process described in detail. In the final design all class diagrams are drawn again and by using these diagrams and their definitions.

3. A prototype for user interface parts of Akamai Web Toolkit is designed to make clear for the web developer users. This prototype is also helped us to implement our database unit and architecture unit more carefully and extend it new and powerful features. In the final design phase of this project we have given a more detailed
user interface design which is refined and possibly extended with other important features.

5. File formats described in initial report has been refined again and last versions will be prepared in final design phase of Akamai Web Toolkit Project.

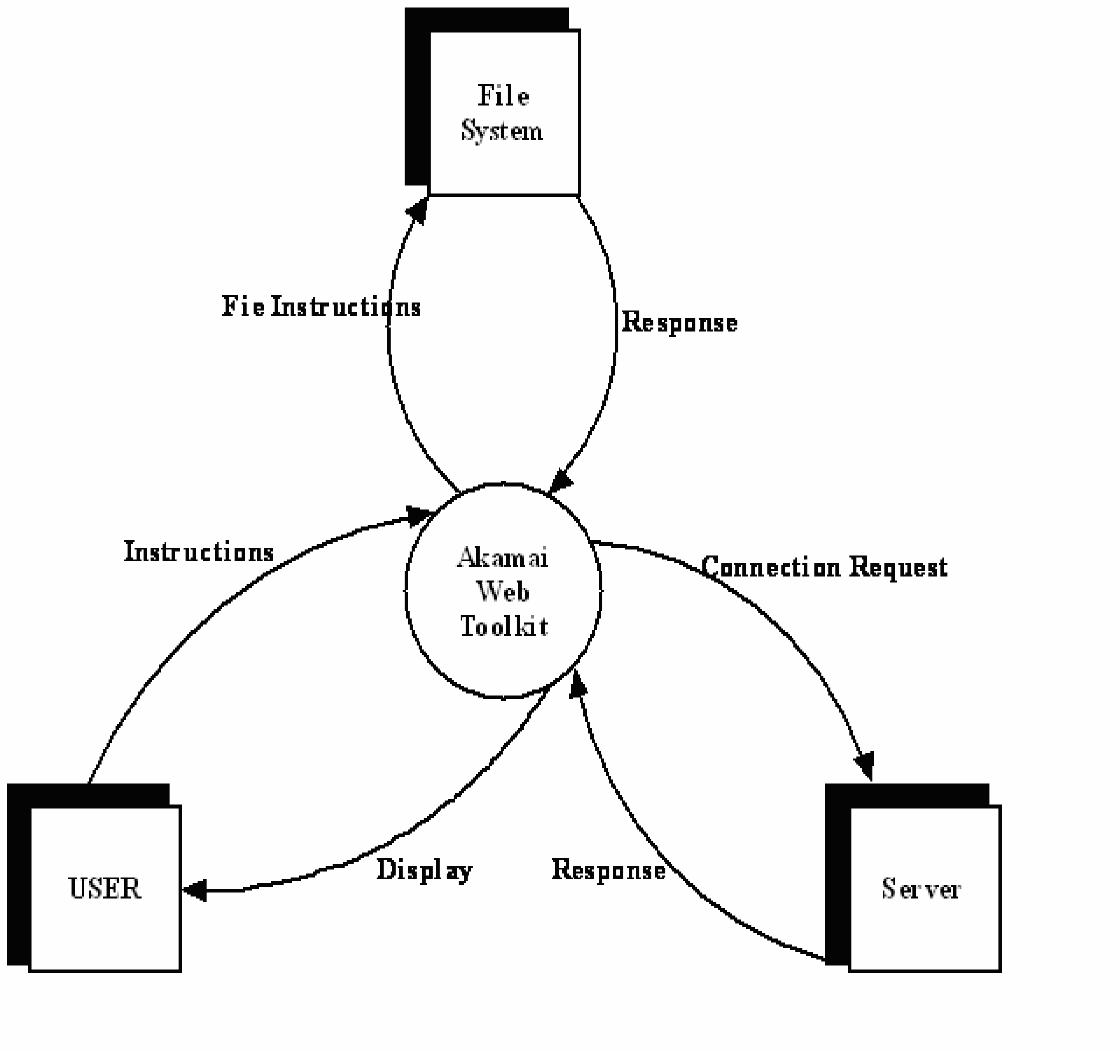
6. Coding specifications and Overall Software Documentation is prepared during final design while working on class diagrams of the components in the architecture.

2. SYSTEM DESIGN

2.1 Data Flow Diagrams:

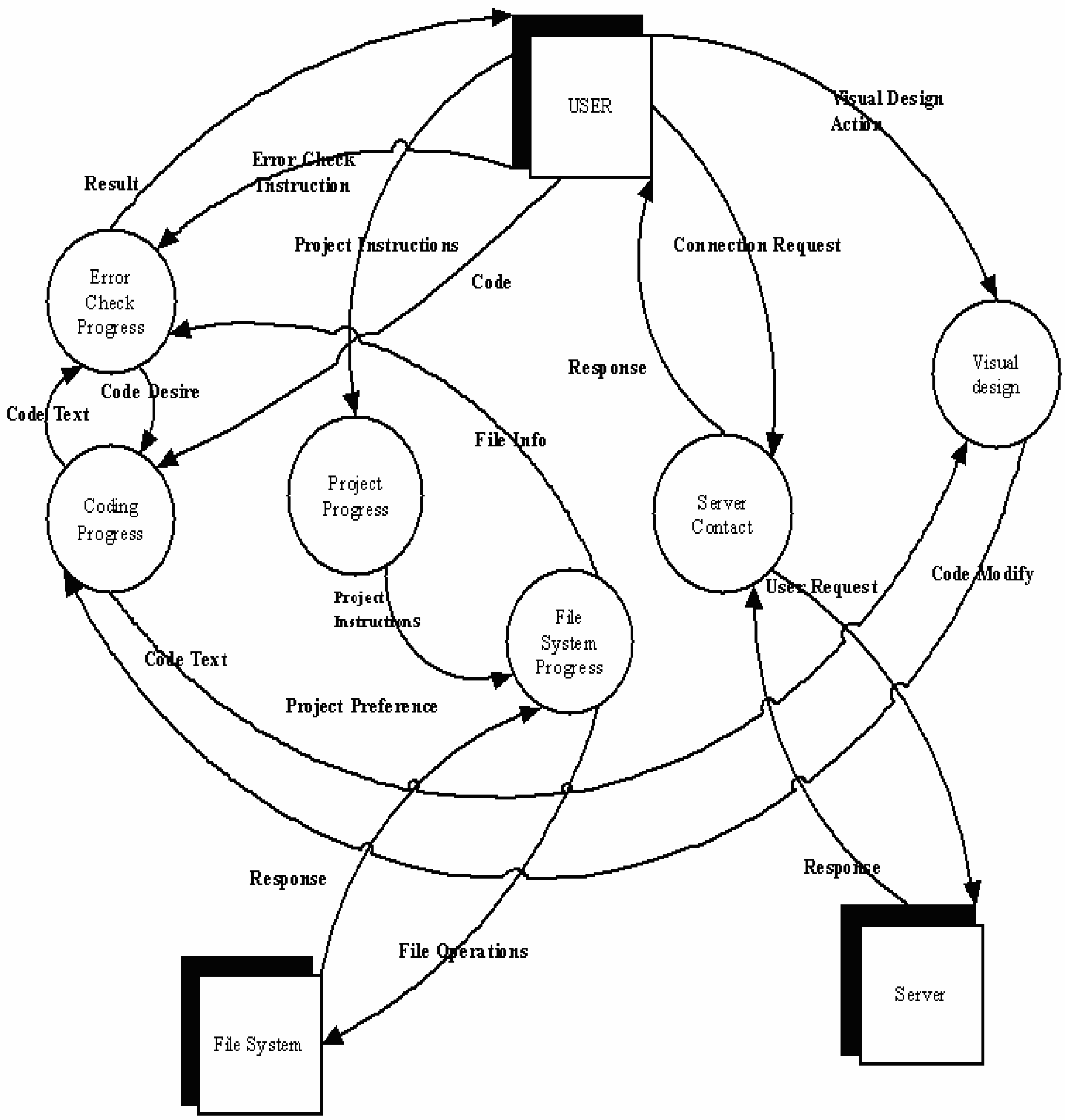
In this section, the functional model of Akamai Web Toolkit is presented. It is composed of process specifications and DFD of the five different levels (Level0, Level1, File System, Error Check and Display) of the web development.

Level 0:



As seen from the diagram, we have three external entities which are File System, Server and User. The main functionalities of the Akamai Web Toolkit are shown on the diagram. According to the responses and instructions to the File System and Server, user will get the display from the development.

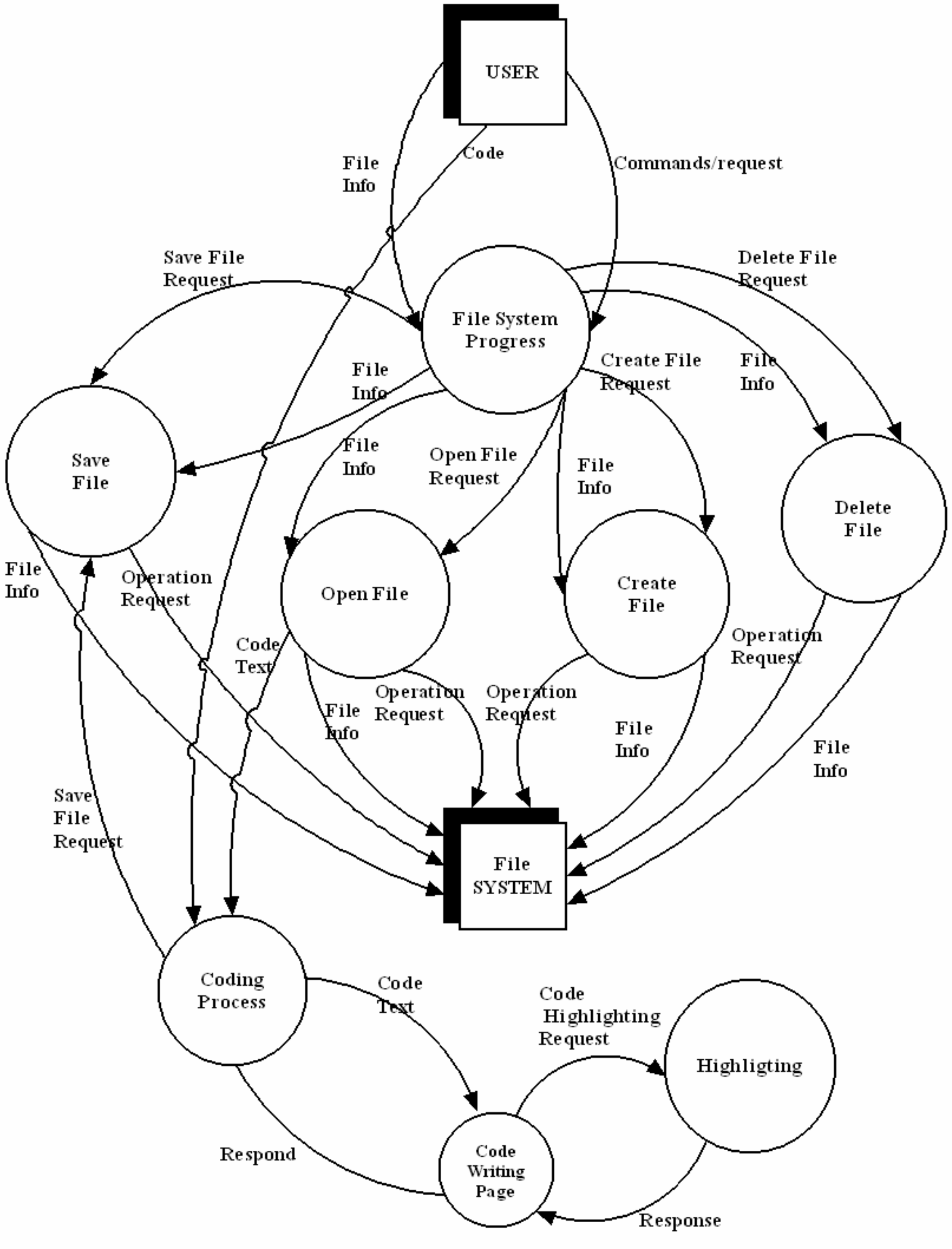
Level 1:



This is a more detailed diagram of Akamai Web Toolkit. This is the most important part because all the requests and responses are processed here. First the requests and responses are parsed and then it is analyzed. As you see in the diagram everything is related with each other. User requests the connection of database from the server and gets responses from there. User also does the coding progress, project progress and error check instructions, while in coding progress user can also are able to use the error check progress while put the code text to the error check progress. While file system progress does the file operations to the file system it will gets responses from there and it will comes

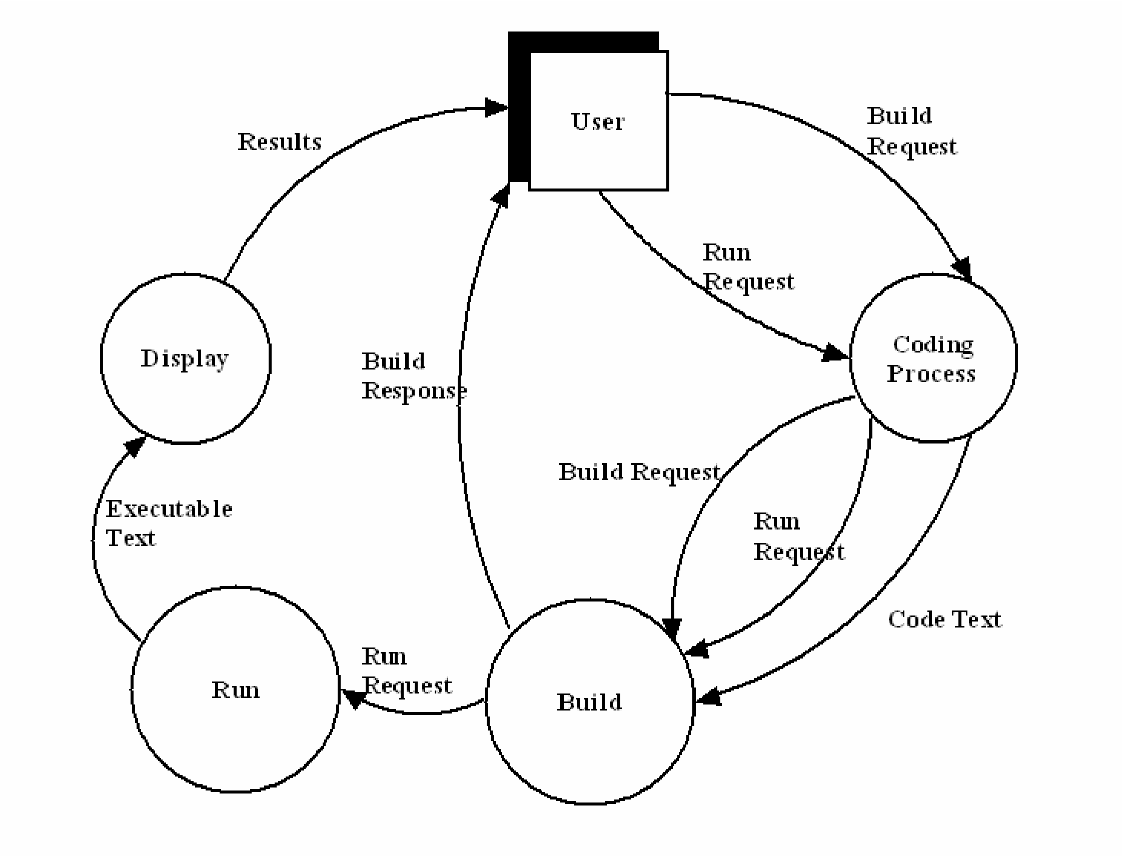
back to the error check progress as file information. At the end user will get the visual design by applying the visual design action. After user gets the visual design they can also be modify code in coding progress and while code progress is related with error check option it will return back to visual design as a right code. .

File System:


USER
Code
File Info
Commands/request
Save File Request
Delete File Request
File System Progress
File Info
Create File Request
File Info
Save File
File Info
Open File Request
File Info
Delete File
Open File
Create File
File Info
Operation Request
Code Text
Operation Request
File Info
Operation Request
Operation Request
File Info
File Info
Save File Request
File SYSTEM
Coding Process
Code Text
Code Highlighting Request
Highligting
Respond
Code Writing Page
Response

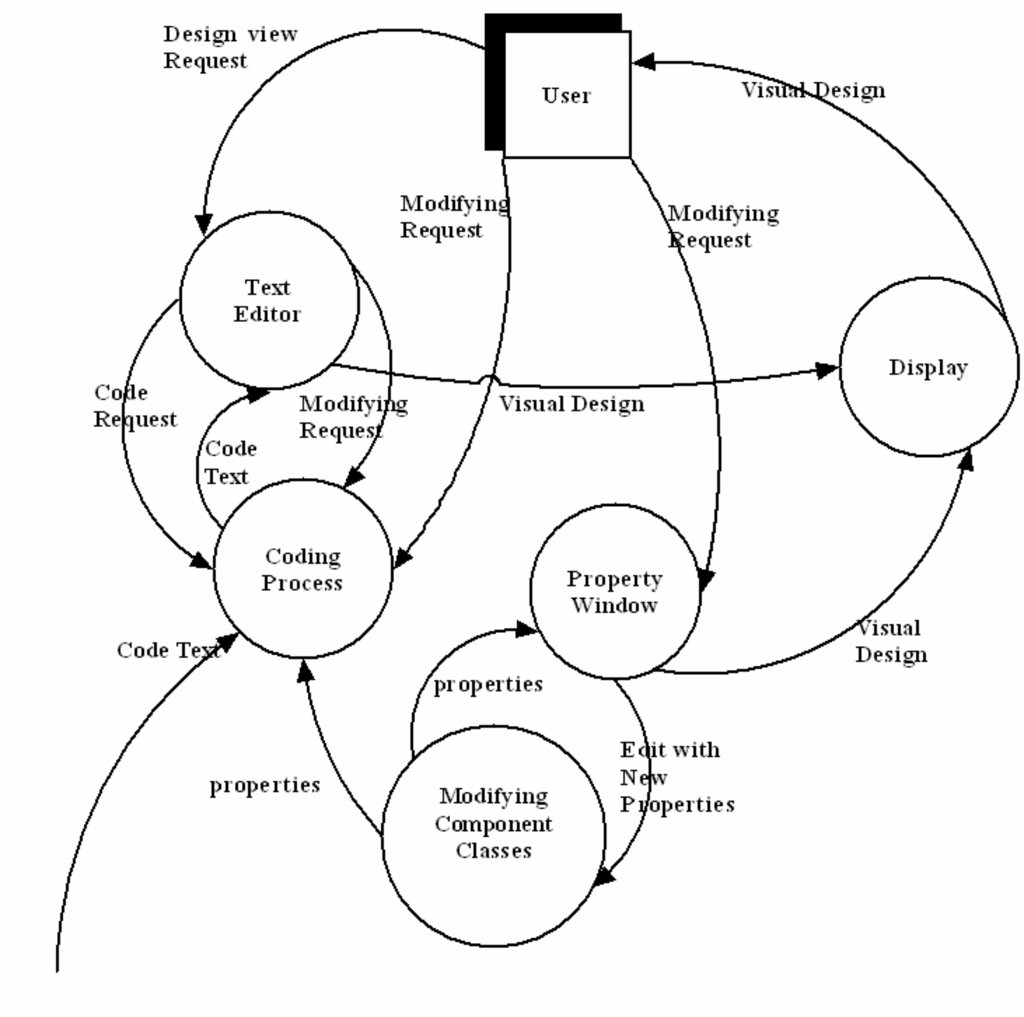
This dfd diagram shows the File System Progress, which is the most complex diagram. This file system progress includes all the operations such as save file, open file, create file and delete file. File System Progress sends to all these operations file requests and these requests are done by File System and returns to the user. While open file is requested by user to be able to write the code and this is done by in coding progress ,while in coding progress user can also see the highlighted texts.

Error Check:



As seen from the diagram, this is the error check data flow diagram. User requests the build and run option in the coding progress and this request will build the code and after this user can be able to see the errors. Requesting the run option is executes the code and after that user can see the display screen and this will returns back to the user as a result.

Display:



This dfd diagram is for display of Akamai Web Toolkit. User will request to modify the property window and coding process and also request a design view from the text editor. All these requests returns back to the user as vision. Property window request to edit with new properties from modifying component classes and this properties returns back to coding progress and from here it returns back to the text editor and from text editor it display the vision for user.

2.2 Data Dictionary:

Name	Edit with new properties
Where and How Used	Modifying component class
Description	Editing components

Name	Properties
Where and How Used	Property window
Description	Editing properties of components

Name	Results
Where and How Used	Display
Description	Showing the page

Name	Run request
Where and How Used	Run
Description	Display page in browser

Name	Build request
Where and How Used	Coding process
Description	To check whether it is ready to execute or not

Name	Executable Text
Where and How Used	Display
Description	Code to be run

Name	Operation request
Where and How Used	File system
Description	Open file

Name	Code highlighting request
Where and How Used	Code highlight
Description	Displaying tags in different colors

Name	File Operations
Where and How Used	File system
Description	Adding file, deleting file, editing file

Name	File Info
Where and How Used	File system progress
Description	Info of file that will pass to file operations

Name	Error check instruction
Where and How Used	Error check
Description	User’s request to error check, build

Name	Code Text
Where and How Used	Coding Progress
Description	Html, java script or css text

Name	Project Instruction
Where and How Used	Project Progress
Description	Add file, close project

Name	Visual Design Actions
Where and How Used	Visual Design
Description	Adding components, editing components

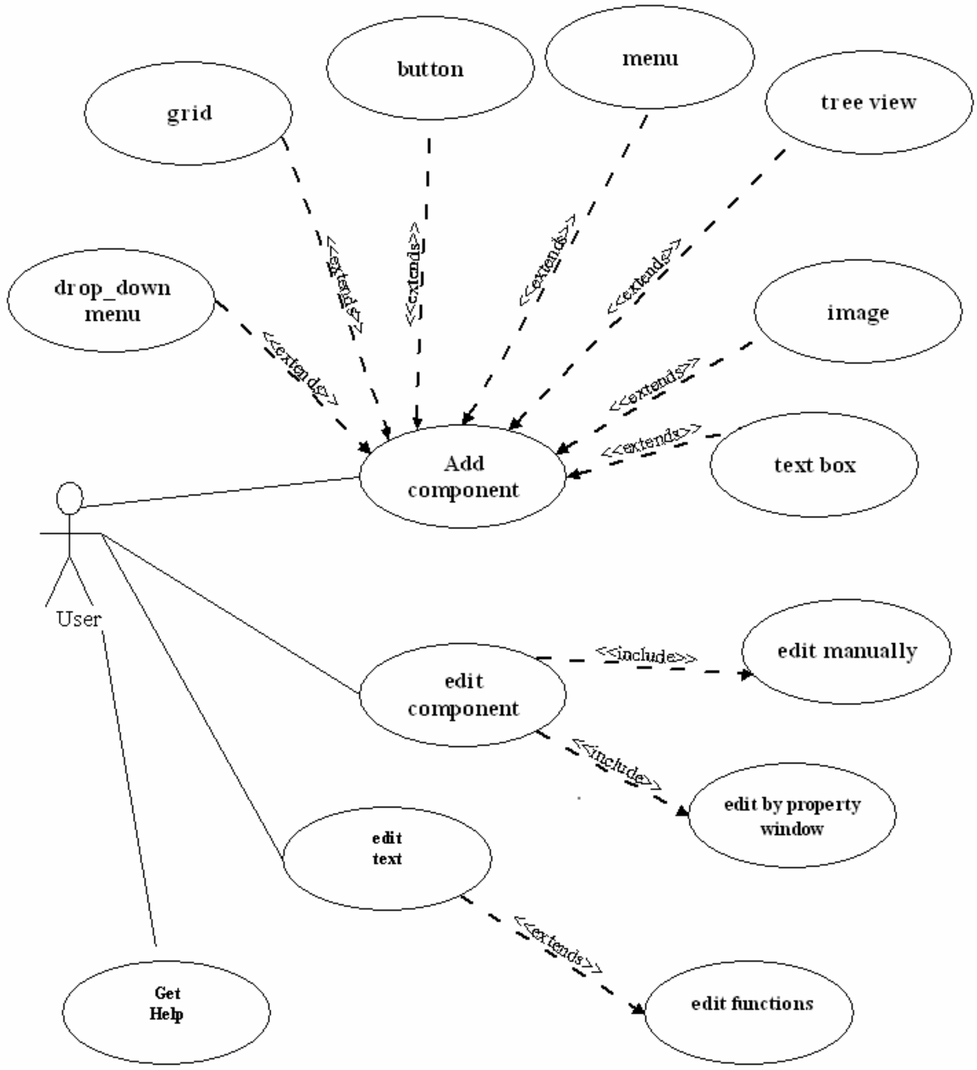
Name	Connection Request
Where and How Used	Database Connection
Description	Connection request to server or database

Name	Code Modify
Where and How Used	Coding Process
Description	Editing source, writing scripts

2.3 Use Case Diagrams

When user starts program, a start page will be shown. If it is the first time for executing program, a welcome page will be shown to the user. In the program directory, there will be a document which keeps recent project names. In this page user can find shortcuts to the recent projects. In order to open an old project, user should select related ".wtk" file for the project. If user wants to start a new project, then he/she clicks the new button, or first clicks the file from the menu, then clicks new.

Add Component:

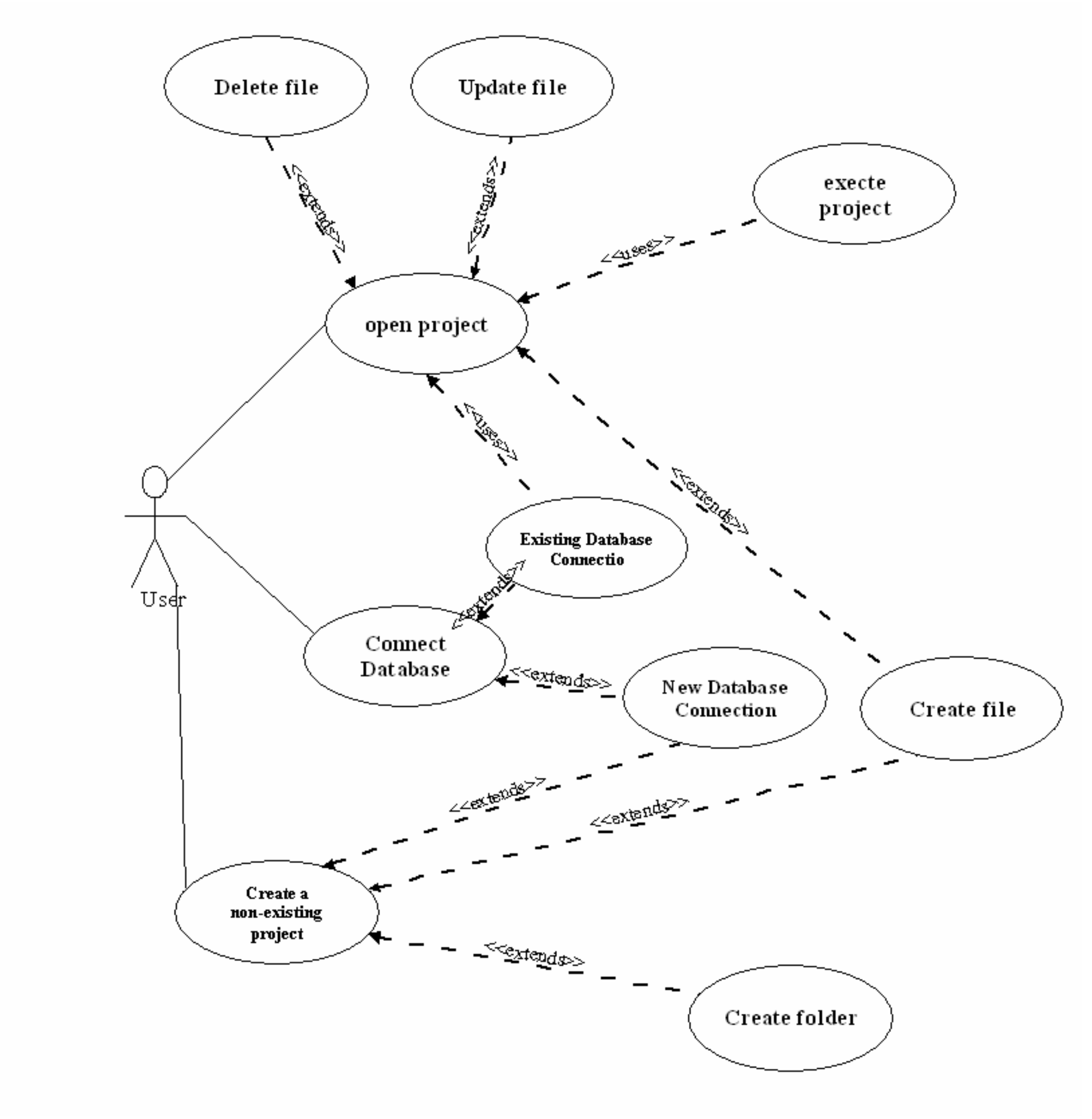


User can add a new file or existing file. If it is a new file, program creates it into the project directory. A new file can be a css file, js file or html file. If user tries to add an existing file, program copy the selected file into the project directory from its current position. In this option, user can add any type of file. User can also create new folders in the project directory. These files can be seen in solution explorer.

User can edit the properties of a component in two ways. Firstly, he/she can edit by editing the source code, and secondly by property window. When user selects the component, in the property window, properties of selected component will appear. When user changes any of them, it will also change in the source. Therefore the changes can be seen immediately. If user edits source code, effects of changes can be seen in the preview mode. This preview mode also works for property window.

In addition to this, since we have a text editor, all operations like cut, copy, undo etc. will be in the program. In other words all text editing operations take place in the program.

Open Project:



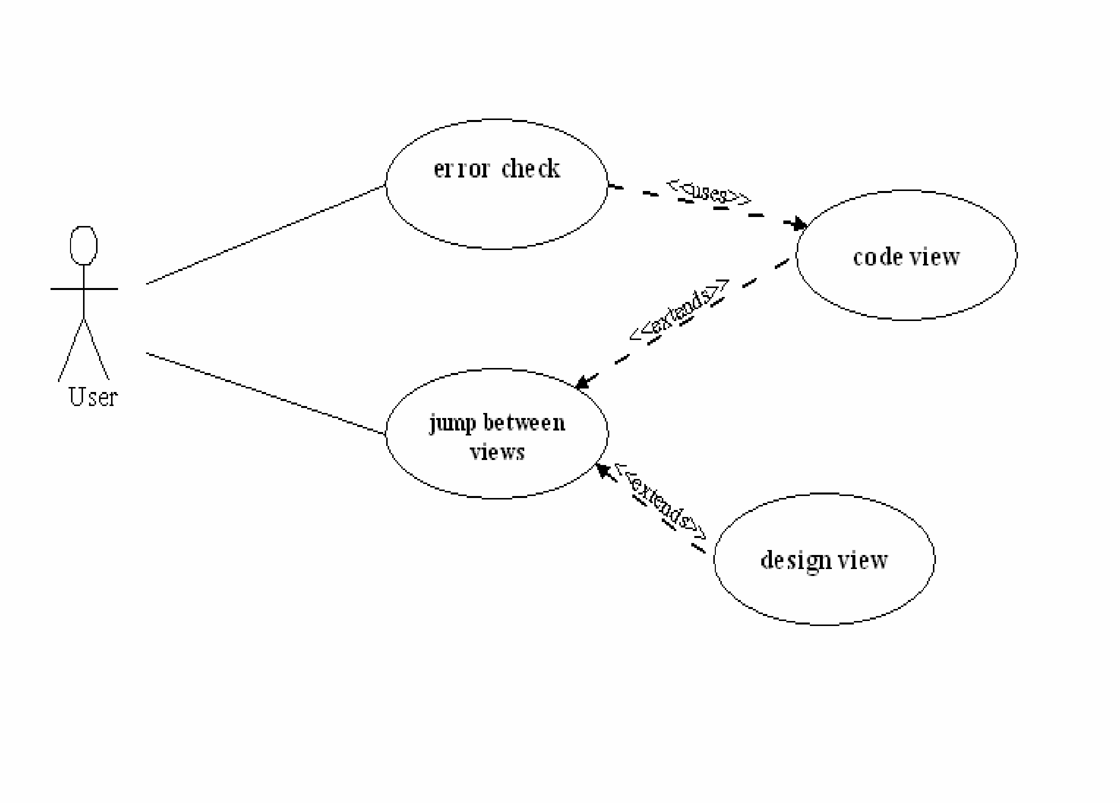
User can open new project, or already created one. If it is a new project, program let user to choose directory to create the project folder and project name. After specifying these, program creates a file for keeping project information in the project directory. This file has project name and an extension "wtk" which is made up by our team and stands for "Web Tool Kit". It contains the file names in the project directory. Next time user opens the project, in the solution explorer the files that are in the project will appear. This file also helps to keep other necessary information that might be occurred. In other words, user should select this file in order to open an old project. When user tries to deploy the project he/she doesn't need to put this file to folder that website is published. If this file is corrupted or deleted, user opens new project and adds the existing files and continues development.

User chooses the database type such as access, ms sql, mysql. If he/she selected access, directory list will be shown to user in order to make him/her select the directory that contains access file. For other databases, user will be directed to select appropriate server. After selecting database, program will create a connection.js file and place "script" tag in html file and connect to the database. Content of the database will be shown in the server explorer window. User will be able to select any table from database and connect it to a component. Java script code that will be used for these operations will also be placed in the connection.js file. User can see the data in tables.

Our program also provide query wizard which will help user to create queries by just clicking. Program will have an interface to connect to database and see the content of the database. After selecting the data, program gives the query that will be used for gathering selected data. In other words, program constructs dynamic queries.

User can build the project, or run the project. If he/she runs the project, program first builds it in order to check for errors. If project is error free, it will be displayed in browser, otherwise errors will be listed in the error messages area of interface. In build process, program makes an error check. User may want to test the correctness of the changes.

Error Check:

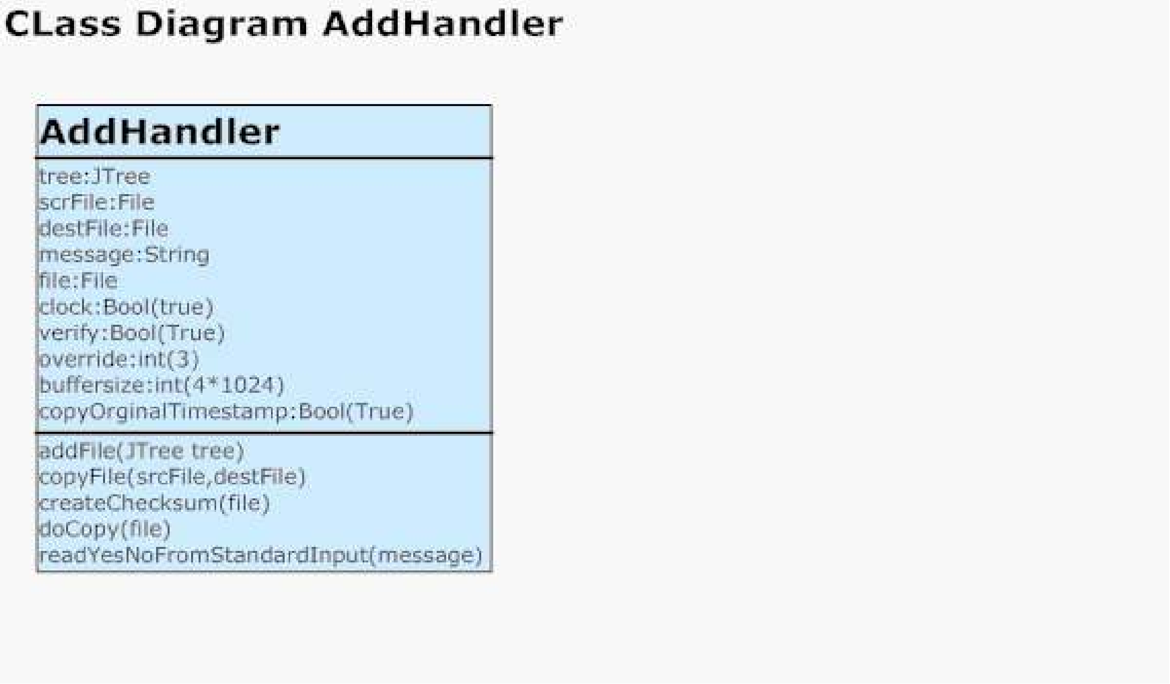


This feature is used for controlling html syntax. When user clicks the build, an error check progress starts. Error check controls whether each tag has closing tag. In addition to this, it controls whether a tag has correct properties. If there is an error, program underlines the incorrect tag in red and shows the error to the user in a tool tip. The tool tip will be shown when mouse comes over the underlined tag.

User can see the design in Mozilla Firefox with preview feature. Any time of development, if user clicks the preview, he/she will see the page as how it will look over the internet. Moreover, user has opportunity to test the application. Testing of the development is also an important part. Checking the connection, commenting on the design, controlling components and their works can be easily done.

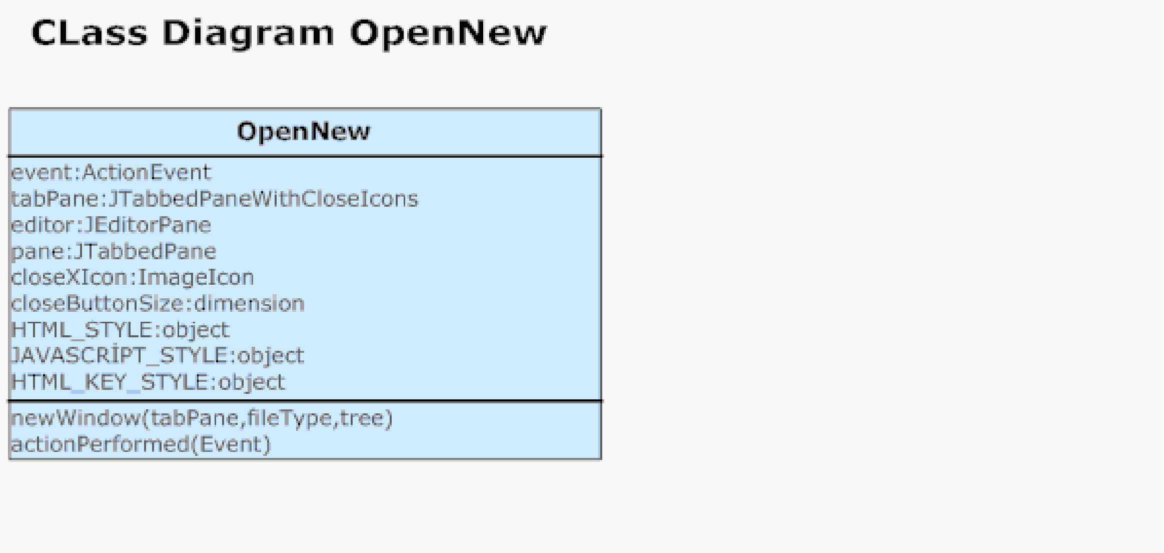
2.4 Class Diagrams

Add Existing File:



This class is used for adding an existing file to the project. User chooses the file and this class copies the file into the project directory. “Add File” item from the “File” menu calls the addFile function. This function opens the file dialog. User chooses the file to be added to project and clicks OK. Then the file is copied to project directory and displayed in the solution explorer.

Add New File:



CLass Diagram OpenNew

OpenNew

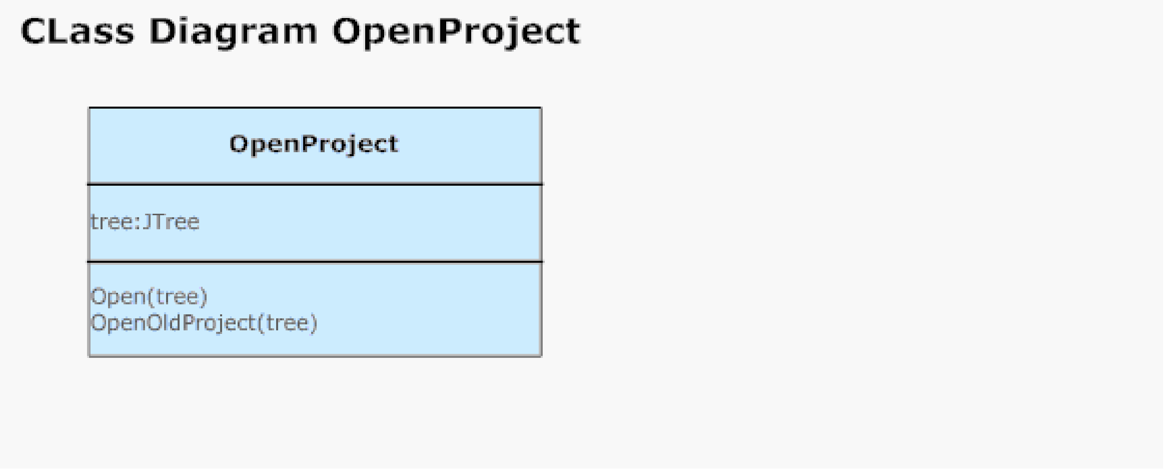
event:ActionEvent
tabPane:JTabbedPaneWithCloseIcons
editor:JEditorPane
pane:JTabbedPane
closeXIcon:ImageIcon
closeButtonSize:dimension
HTML_STYLE:object
JAVASCRİPT_STYLE:object
HTML_KEY_STYLE:object

newWindow(tabPane,fileType,tree)
actionPerformed(Event)

This class is used for creating new files in projects. It creates the desired file and displays it in the interface. This class adds a new tab in the main tab. If desired file is a html file, it adds a new tab in the created tab in order to handle preview. While creating new file, since it displays its content in main tab, it also achieves the highlighting.

We have found an open source text highlighting component. While opening a new file, program sets appropriate highlighting property. It is set by: *document.setHighlightStyle(HTML_STYLE);* sentence. For java script it is “JAVASCRIPT_STYLE”. We couldn’t manage a highlighting for css document.

Open Project:



This class opens either new or existing project. “Open” function creates new project. It displays directory map to user to let them choose the directory and creates project in the selected directory and it sets recent project as this new project. Finally, it displays content in the solution explorer.

“OpenOldProject” function is used for opening an existing project by selecting its “<projectname>.wtk” file. Function shows file dialog. After user selecting the project file, program gets its contents, shows it in the solution explorer and sets recent project as this project.

Add Component:

All components are represented in individual classes. Their attributes are determined according to their properties. All of these classes have their operations as methods. Each of them has add, edit delete, and save methods which get help from html parser class. Thus, it will be easy to keep and change component’s properties.

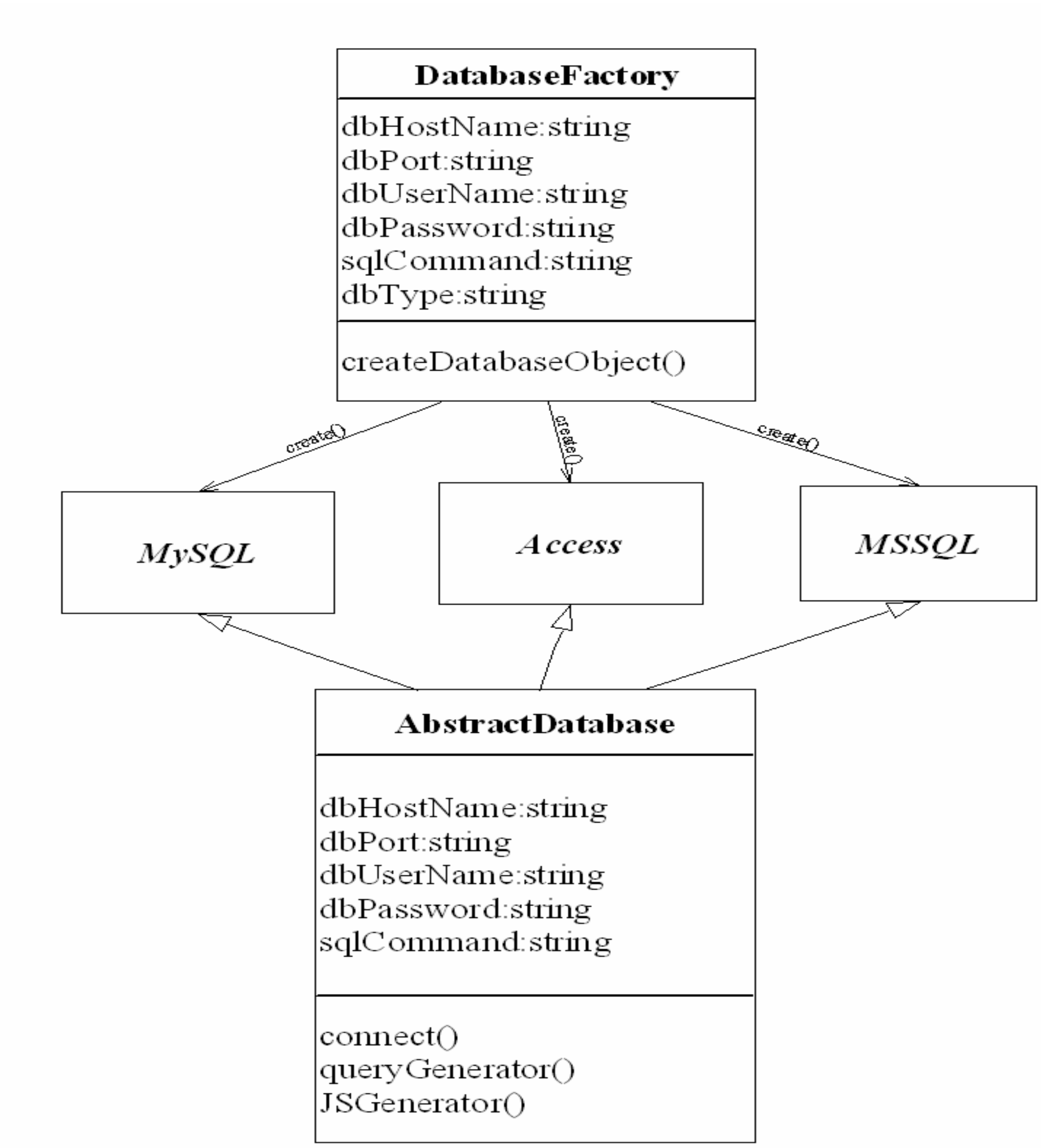
HTML Parser:

Html parser class will be used for parsing html files. With this function, we will be able to find related tags to update, delete or positions to insert tags. Error check property also gets help from this class, because in order to apply an error check, program needs tags individually.

While editing components by property window, selected component’s information is taken to related component’s class in correct type and displayed in the property window. New values are taken after editing. Finally changes are inserted to the source. Tag which is related to selected component is updated. In this feature, program does the changes in code instead of user. User just interacts with interface, and program makes the desired changes. Class' edit method will be used for this operation.

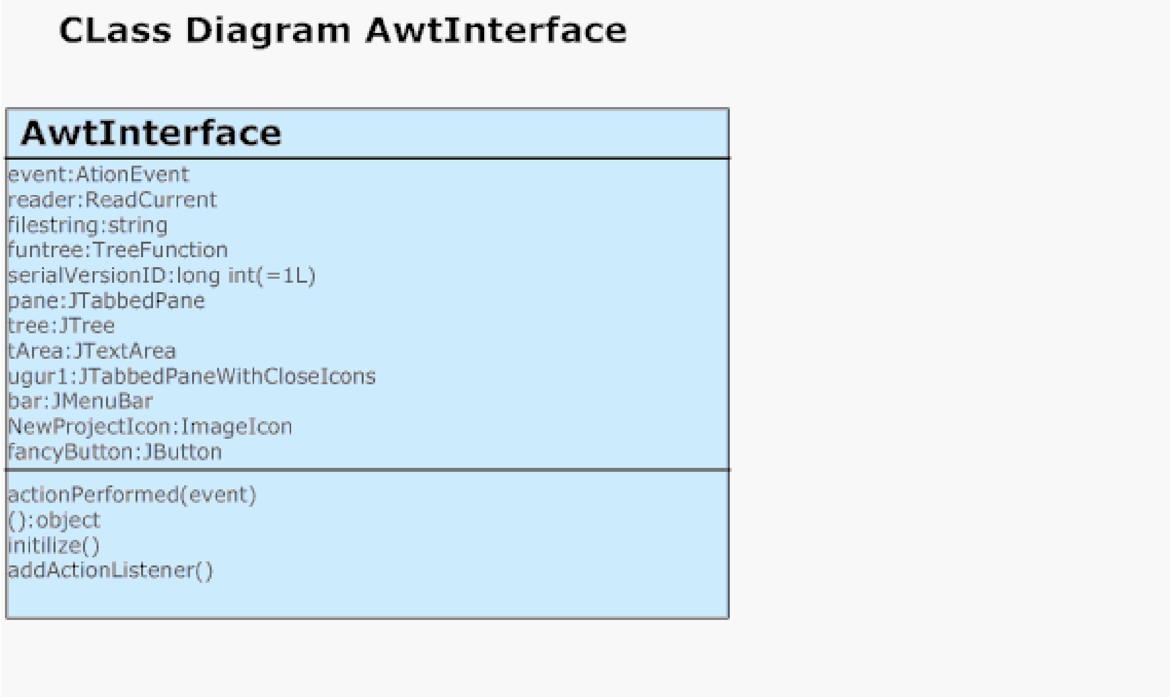
In order to make editing on html files, we will code an editing library which is used for file operations. We need to parse files and take necessary parts. After editing, we should insert the updated lines. For these purposes, we write a file editing library. Main part of this library will be html parser.

Database:



There is an abstract database class. This class keeps common properties of databases. All databases have connection strings, sql commands to execute and operations such as connecting, creating command, and java script creation. These methods will be implemented for each database. There is a database factory class which creates the correct database objects. In order to execute the given sql command, database needs a connection. For our query wizard, query generator function will be called. Java script code for connecting to database will be generated by jsgenerator function. These functions must be very strong background in order to meet all requirements. This kind of dynamic library will be used in later applications.

Akamai Web Toolkit GUI:



CLass Diagram AwtInterface

AwtInterface

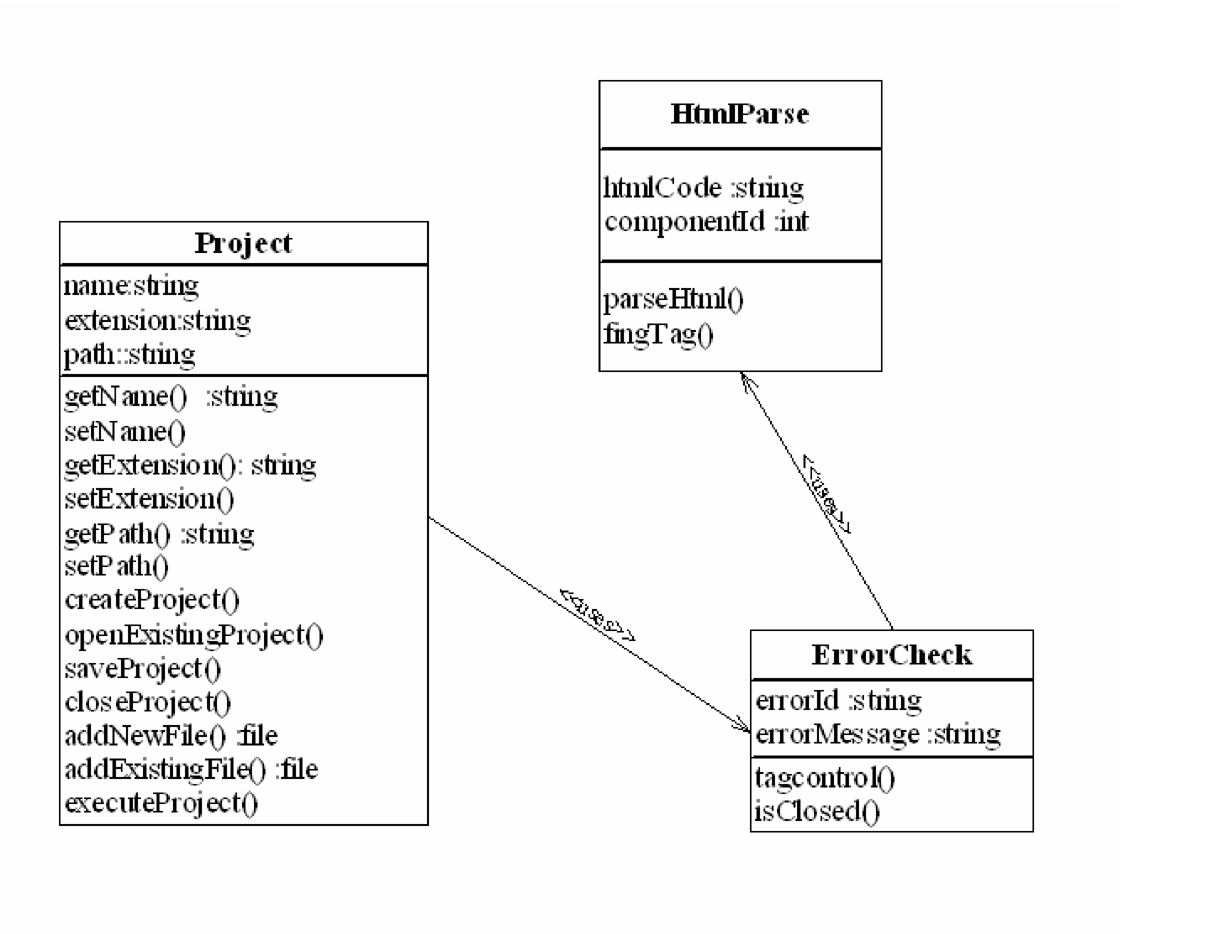
event:AtionEvent
reader:ReadCurrent
filestring:string
funtree:TreeFunction
serialVersionID:long int(=1L)
pane:JTabbedPane
tree:JTree
tArea:JTextArea
ugur1:JTabbedPaneWithCloseIcons
bar:JMenuBar
NewProjectIcon:ImageIcon
fancyButton:JButton

actionPerformed(event)
():object
initilize()
addActionListener()

This class is used for creating program's interface. It contains all menu items, panels. And it also contains action listeners of these items. It also opens the last project that the user worked on most recently and displays its contents in the solution directory.

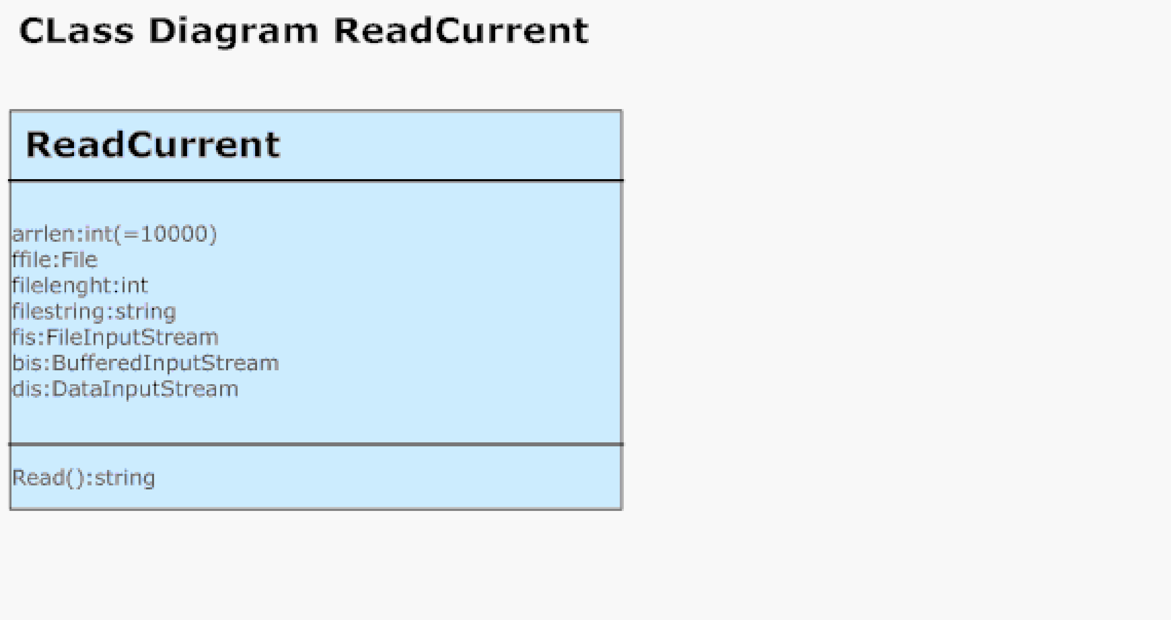
When user wants to see the page developed, clicks preview and program switches to preview mode in which user can see the changes. Thus, user can see the page and make necessary corrections. For this purpose, user may run the project.

Error Check:



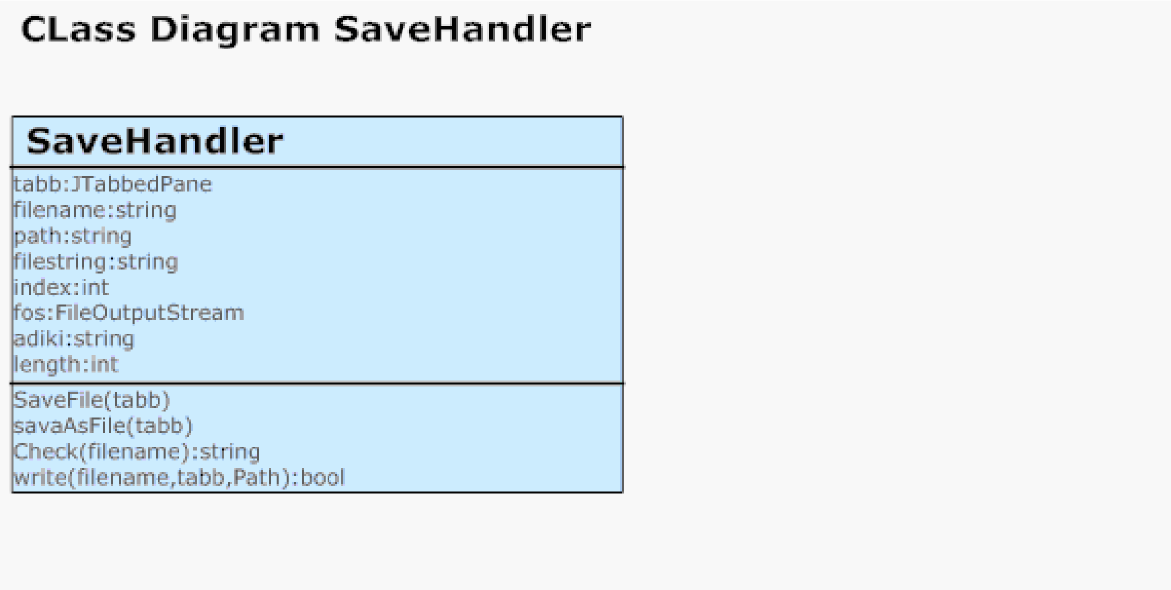
This class controls the tags and raise errors if there are any. Html parser class sends the tags to tag control, and then this method controls tag's properties and syntax. In addition to this, the architecture of html will be built to check whether each tag has closing tag. A xml architecture may be built.

ReadCurrent:



This class is used for reading “current.drc” file. As mentioned, this file keep track of most recent worked project directory. While opening program, this class called and most recent worked project is opened.

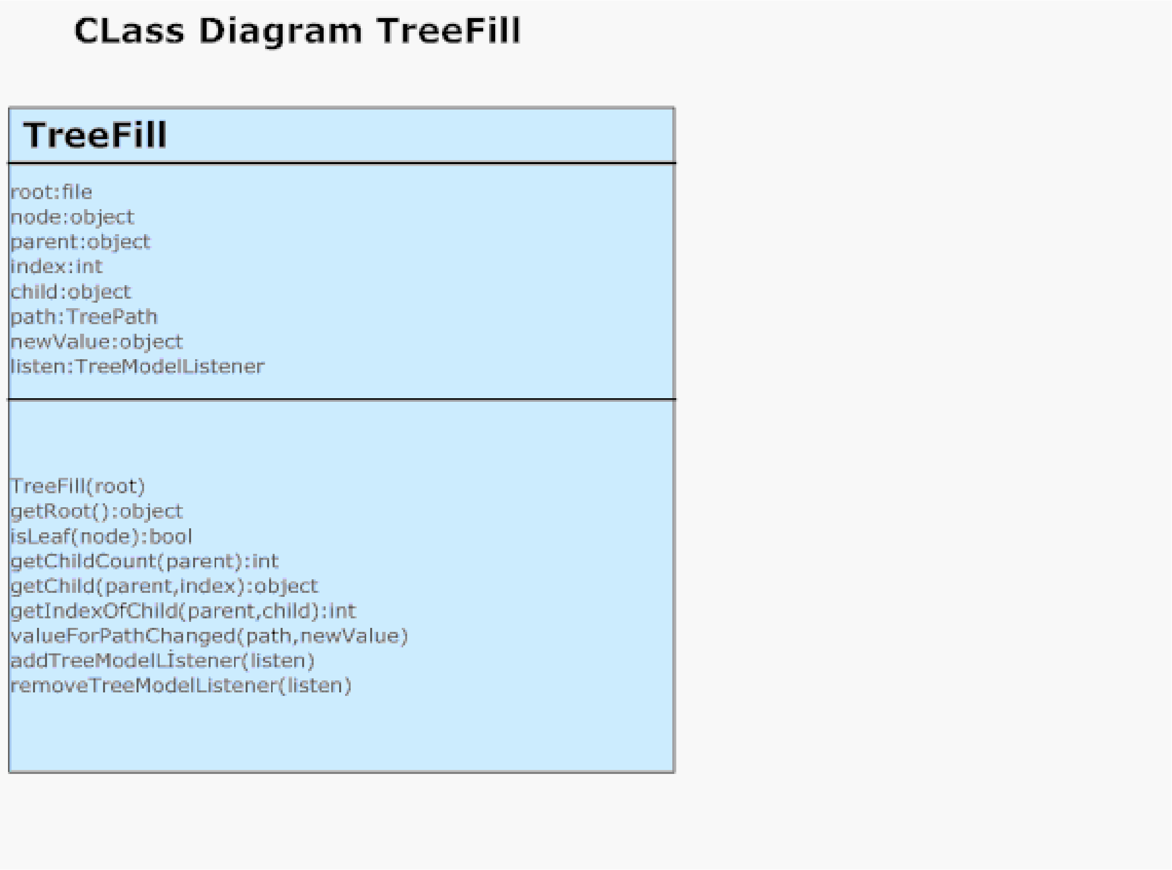
SaveHandler:



This class is used for save operations. There are two main operations. One of them is basic save in which user clicks save and content of file is written to the specified file. In this option the file is saved to where it is created. Second one is “save as” operation. In this option, user selects the path to save file and specifies its name and extension.

Since we open html file in a new tab in the main tab, we did some adjustments in order to reach its content.

TreeFill and TreeFunction:

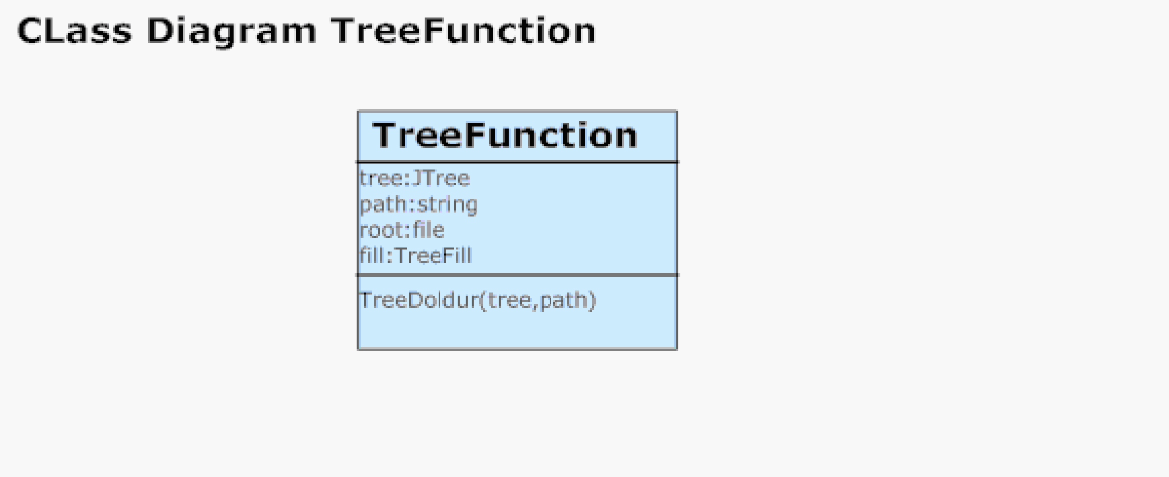


CLass Diagram TreeFill

TreeFill

root:file
node:object
parent:object
index:int
child:object
path:TreePath
newValue:object
listen:TreeModelListener

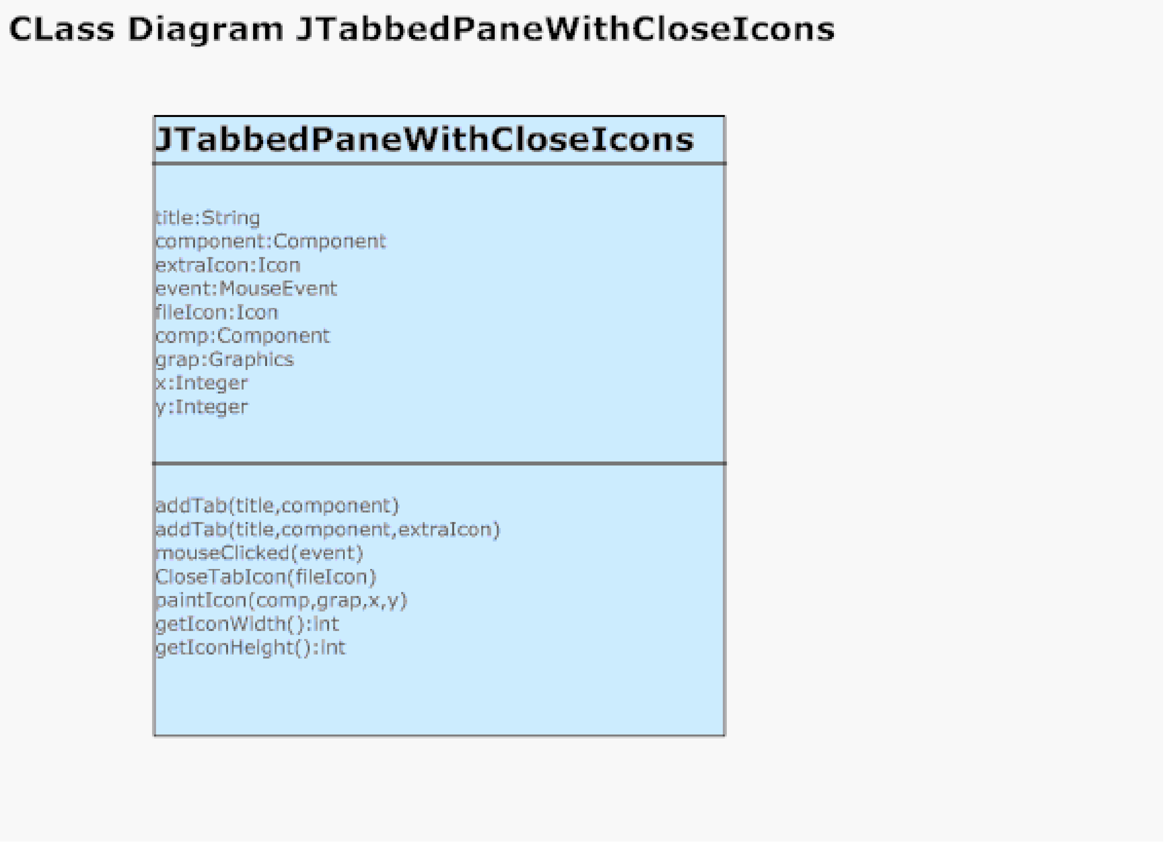
TreeFill(root)
getRoot():object
isLeaf(node):bool
getChildCount(parent):int
getChild(parent,index):object
getIndexOfChild(parent,child):int
valueForPathChanged(path,newValue)
addTreeModelListener(listen)
removeTreeModelListener(listen)



In order to list the project directory in a tree view, we use tree model concept. In this concept, we create a model and just set the model property of tree as this model. This model just needs the root directory, after giving the root, it creates model recursively. Then in the treefunction class, this model is constructed and set the given tree.

While displaying a file, we create an instance of this object (HighlightedDocument) and copy its content to provided text area. Therefore highlighting is automatically created.

JTabbedPaneWithCloseIcons Class:



CLass Diagram JTabbedPaneWithCloseIcons

JTabbedPaneWithCloseIcons

title:String
component:Component
extraIcon:Icon
event:MouseEvent
fileIcon:Icon
comp:Component
grap:Graphics
x:Integer
y:Integer

addTab(title,component)
addTab(title,component,extraIcon)
mouseClicked(event)
CloseTabIcon(fileIcon)
paintIcon(comp,grap,x,y)
getIconWidth():int
getIconHeight():int

This is a class for adding a close button to the tabbedpane. It extends the JTabbedPane class and allow us to add an ‘X’ button to the left of the tabbed title. And by clicking it we can close the tab.

In this class there are some methods to add new tabs with close icons and also another class to draw the ‘X’ button. The functions are;

--public void addTab(String title, Component component)

This function is used to add a new tab to the tabbedpane. Title is the tab’s name and component is what we want component to add to the tabbedpane.

--public void addTab(String title, Component component, Icon extraIcon)

This function is also used to add a new tab to the tabbedpane. Title and component is used like above function. But in this function you can also add an extra button which you can choose an icon to the left of the title.

--public void mouseClicked(MouseEvent e)

When clicked to the ‘X’ icon on the tabbedpane this function is called and the tab is closed.

--Class CloseTabIcon implements Icon

The class which generates the ‘X’ icon fort the tabs. The constructor accepts an icon which is extra to the ‘X’ icon, so you can have tabs like in JBuilder. This value is null if no extra icon is required. It has private variables called int x_pos, int y_pos, int width, int height and icon fileIcon. And functions called;

--public CloseTabIcon(Icon fileIcon)

This function is used as we described above only when an additional icon is used in the tab.

--public void paintIcon(Component c, Graphics g, int x, int y)

This function uses the coordinates of the title of the tabbedpane and draws an ‘X’ button according to these coordinates.

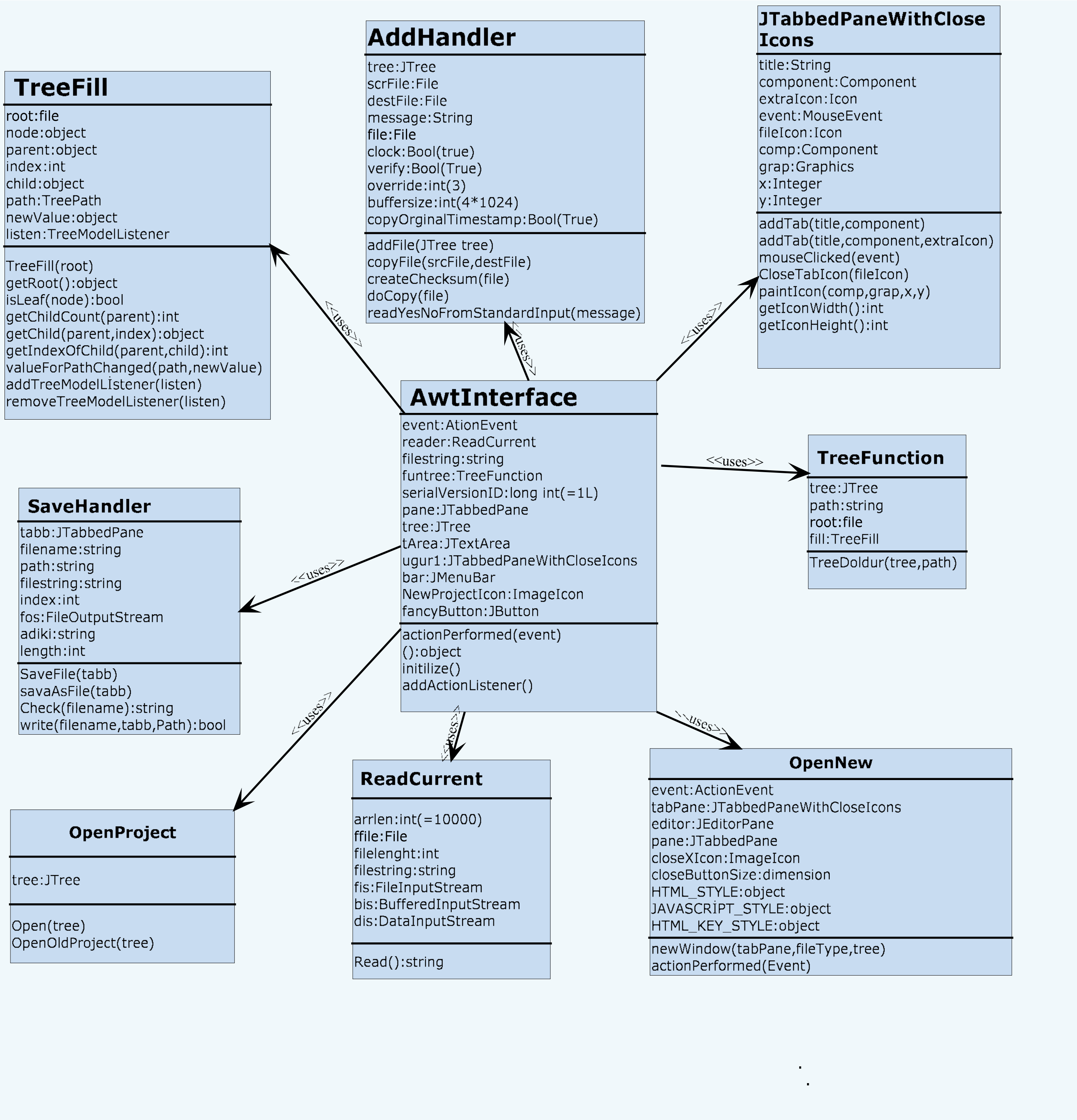
--public int getIconWidth()

This function returns the width of the added icon if there are any added icons.

--public int getIconHeigth()

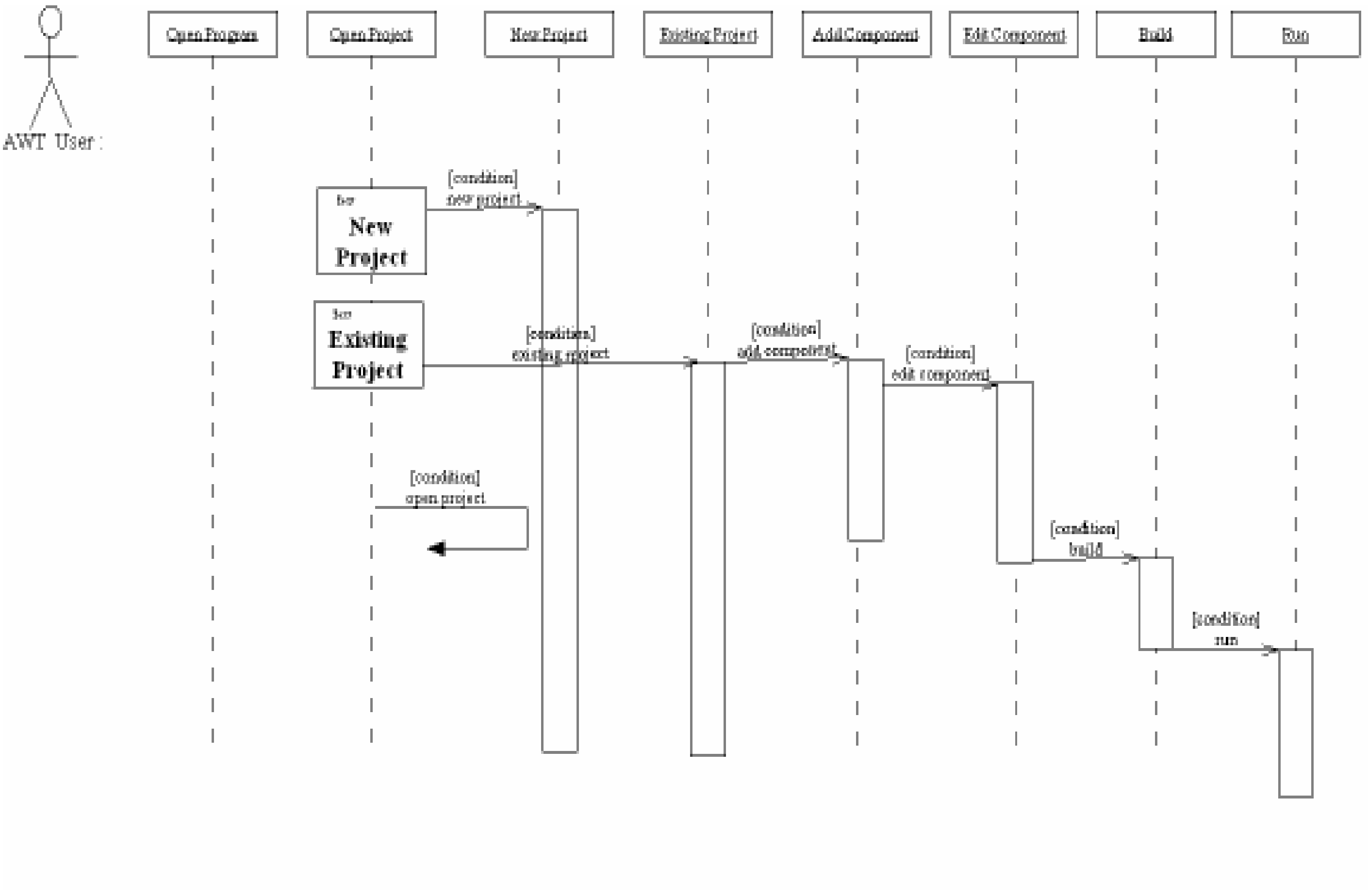
This function returns the heigth of the added icon if there are any added icons.

Overview of System:

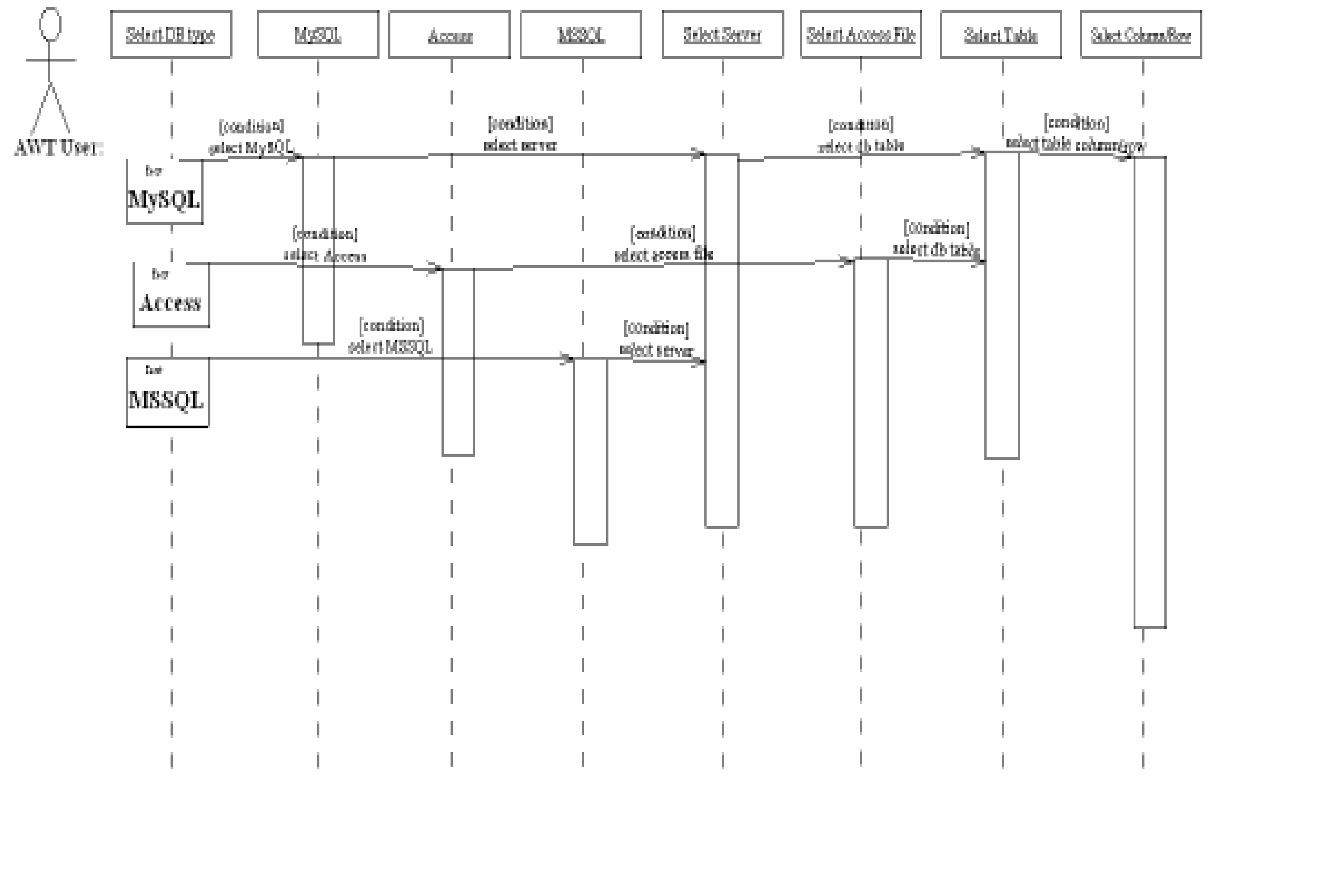


2.5 Sequence Diagrams

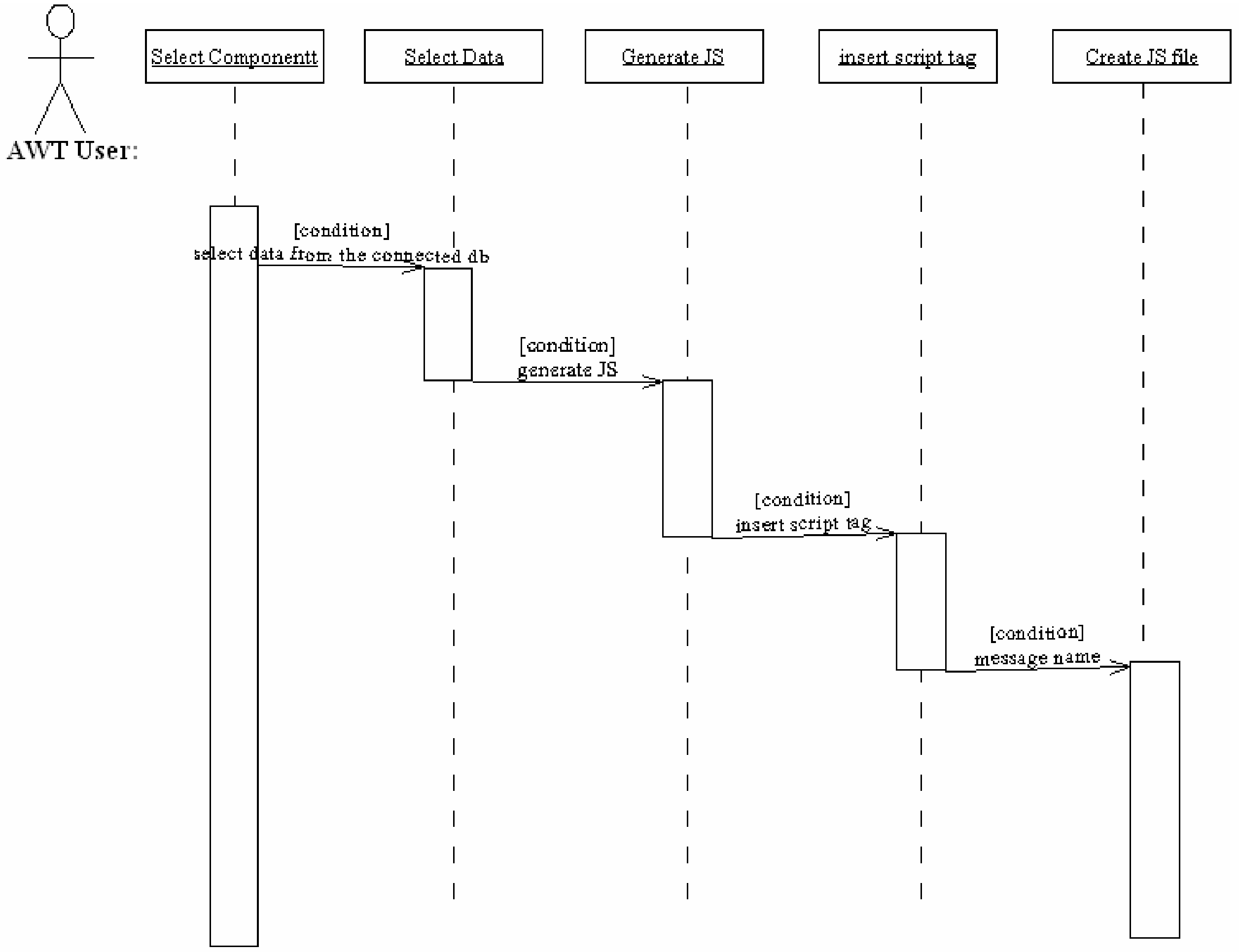
General Flow:



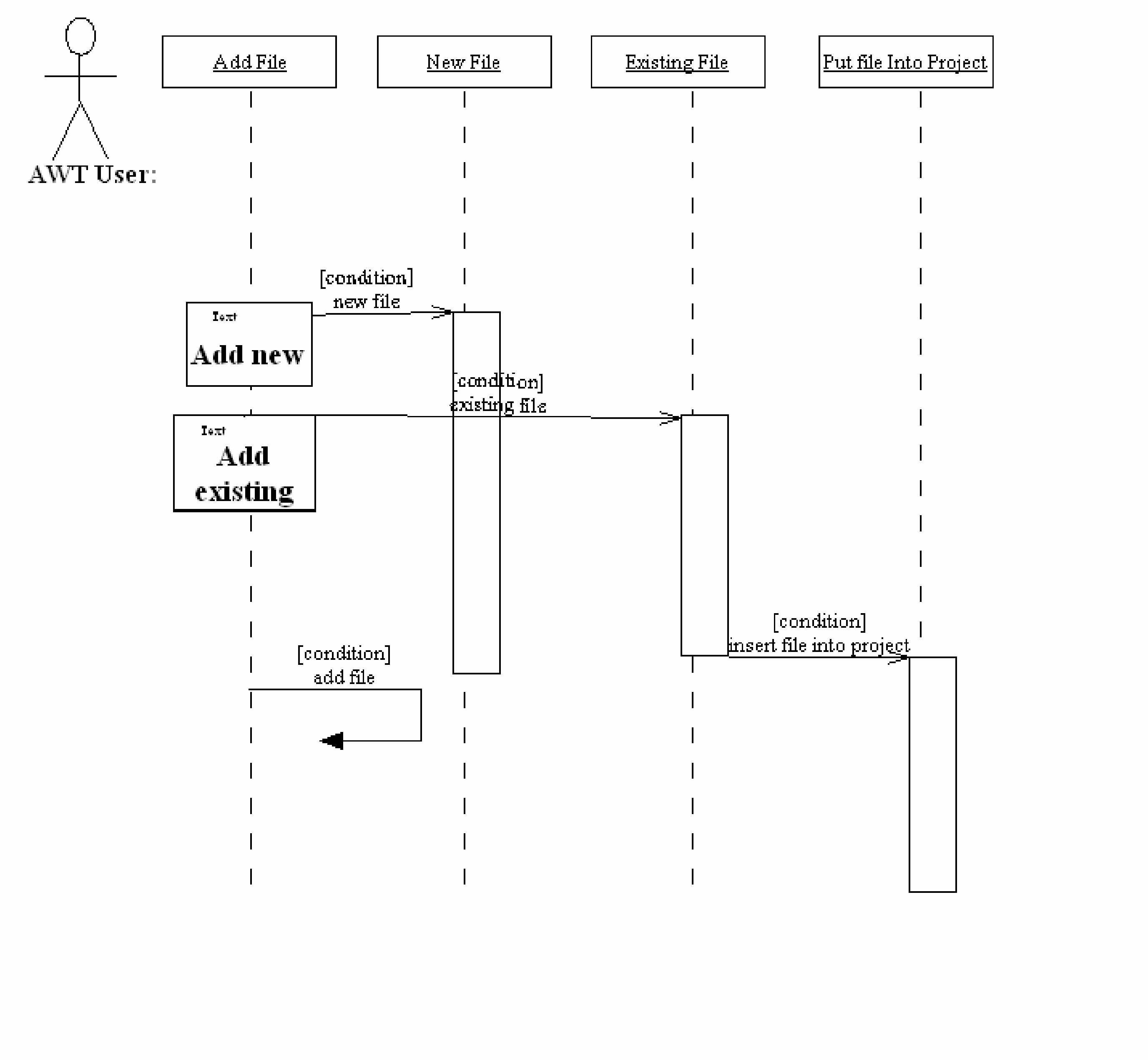
Database Connection:



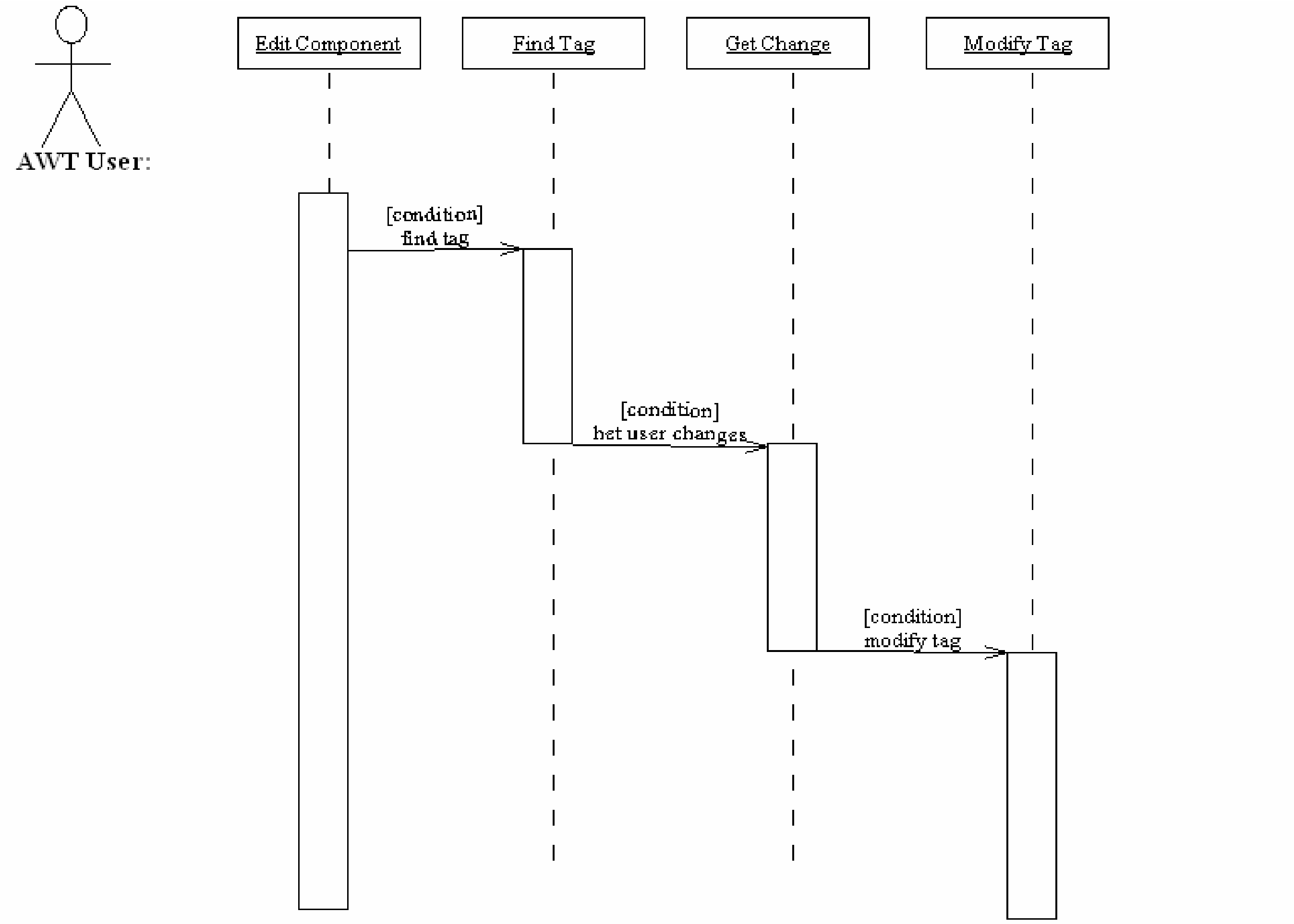
Add action to component:



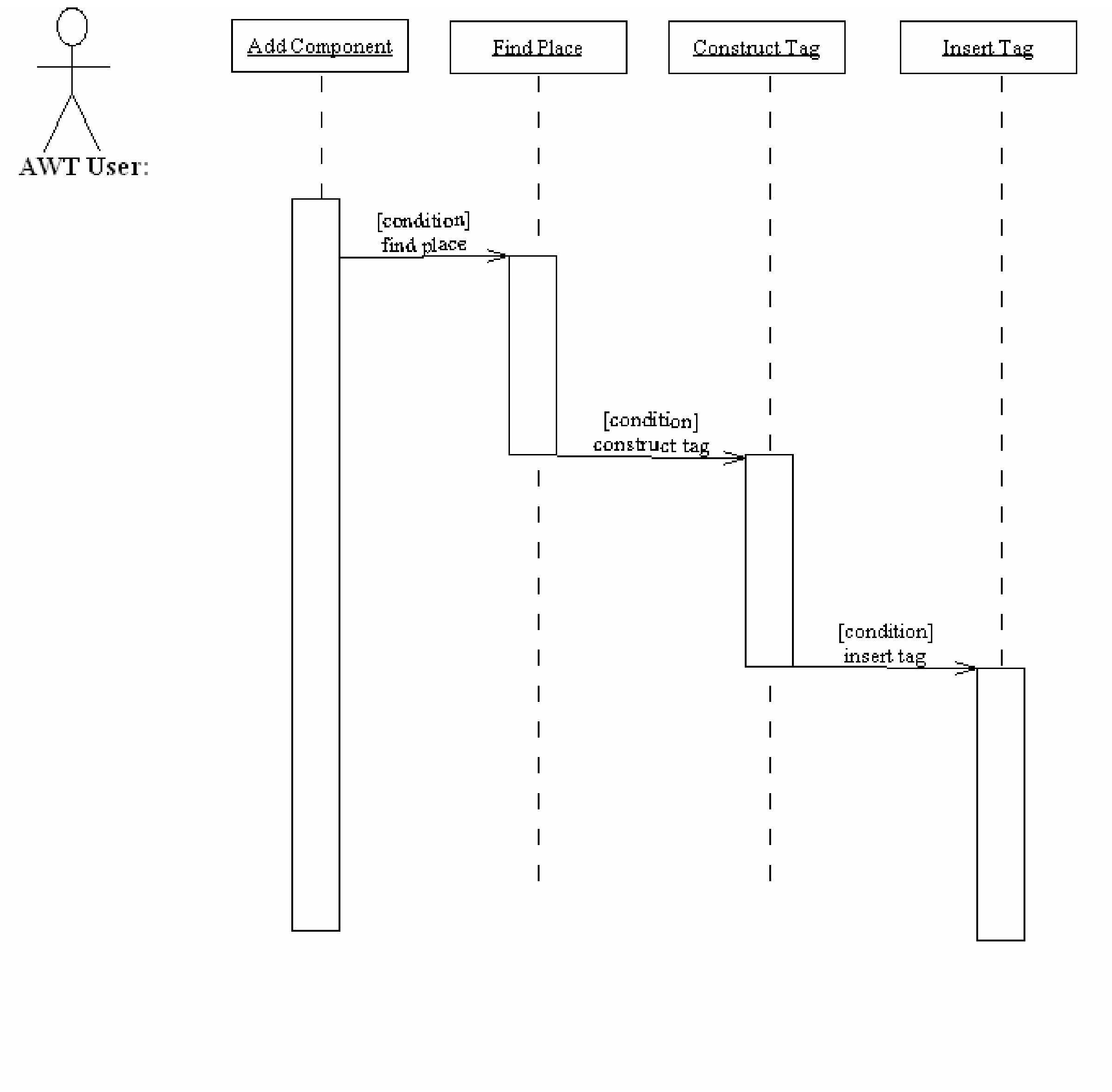
Add file:


AWT User:
Add File
New File
Existing File
Put file Into Project
[condition]
new file
Text
Add new
[condition]
existing file
Text
Add existing
[condition]
insert file into project
[condition]
add file

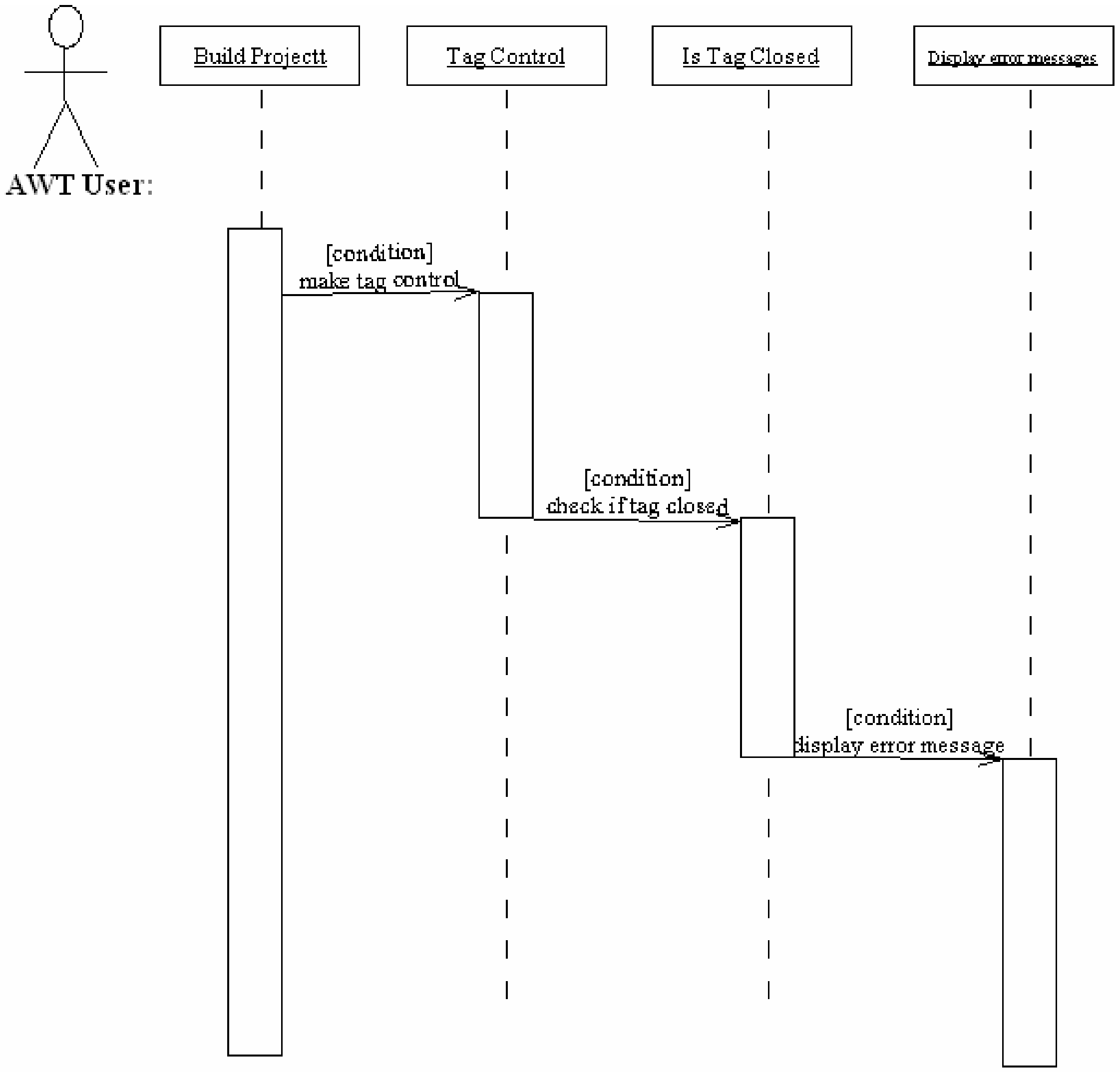
Edit component:



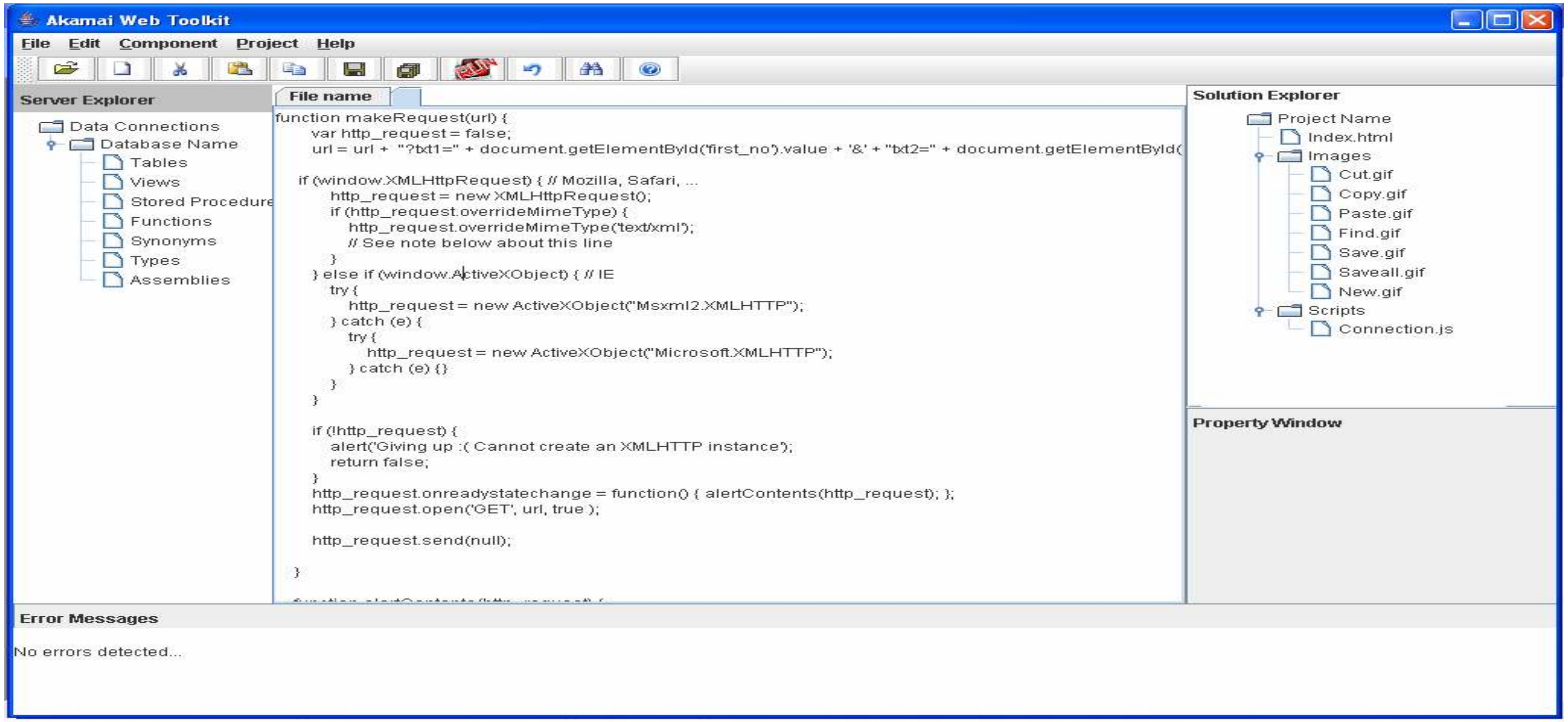
Add component:



Error check:


AWT User:
Build Projectt
Tag Control
Is Tag Closed
Display error messages
[condition]
make tag control
[condition]
check if tag closed
[condition]
display error message

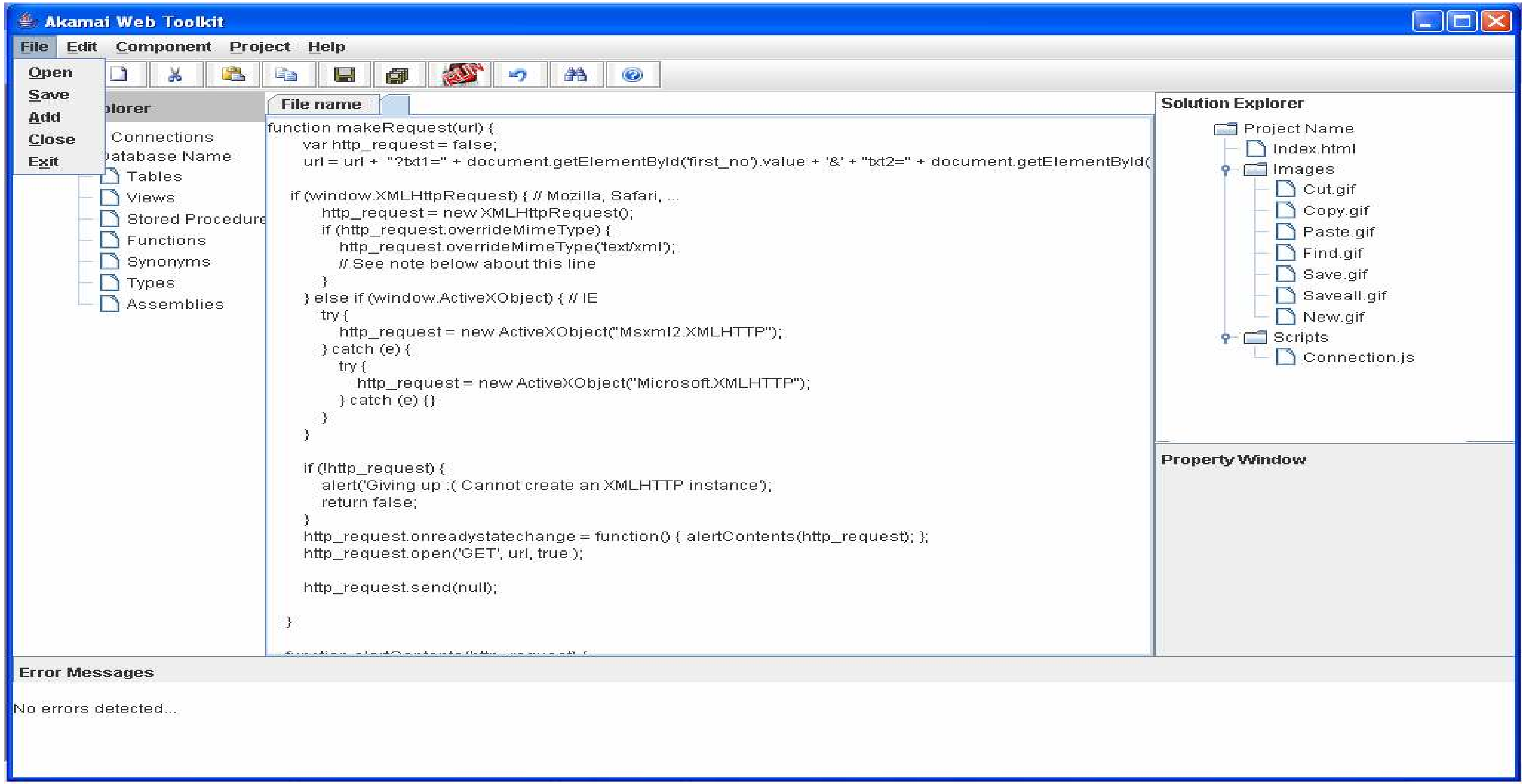
3. INTERFACE



```
function makeRequest(url) {
      var http_request = false;
      url = url +  "?txt1=" + document.getElementById('first_no').value + '&' + "txt2=" + document.getElementById(

    if (window.XMLHttpRequest) { // Mozilla, Safari, ...
         http_request = new XMLHttpRequest();
         if (http_request.overrideMimeType) {
            http_request.overrideMimeType('text/xml');
            // See note below about this line
         }
      } else if (window.ActiveXObject) { // IE
         try {
            http_request = new ActiveXObject("Msxml2.XMLHTTP");
         } catch (e) {
            try {
               http_request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
         }
      }

      if (!http_request) {
         alert('Giving up :( Cannot create an XMLHTTP instance');
         return false;
      }
      http_request.onreadystatechange = function() { alertContents(http_request); };
      http_request.open('GET', url, true );

      http_request.send(null);

   }

   function alertContents(http_request) {
```

No errors detected...

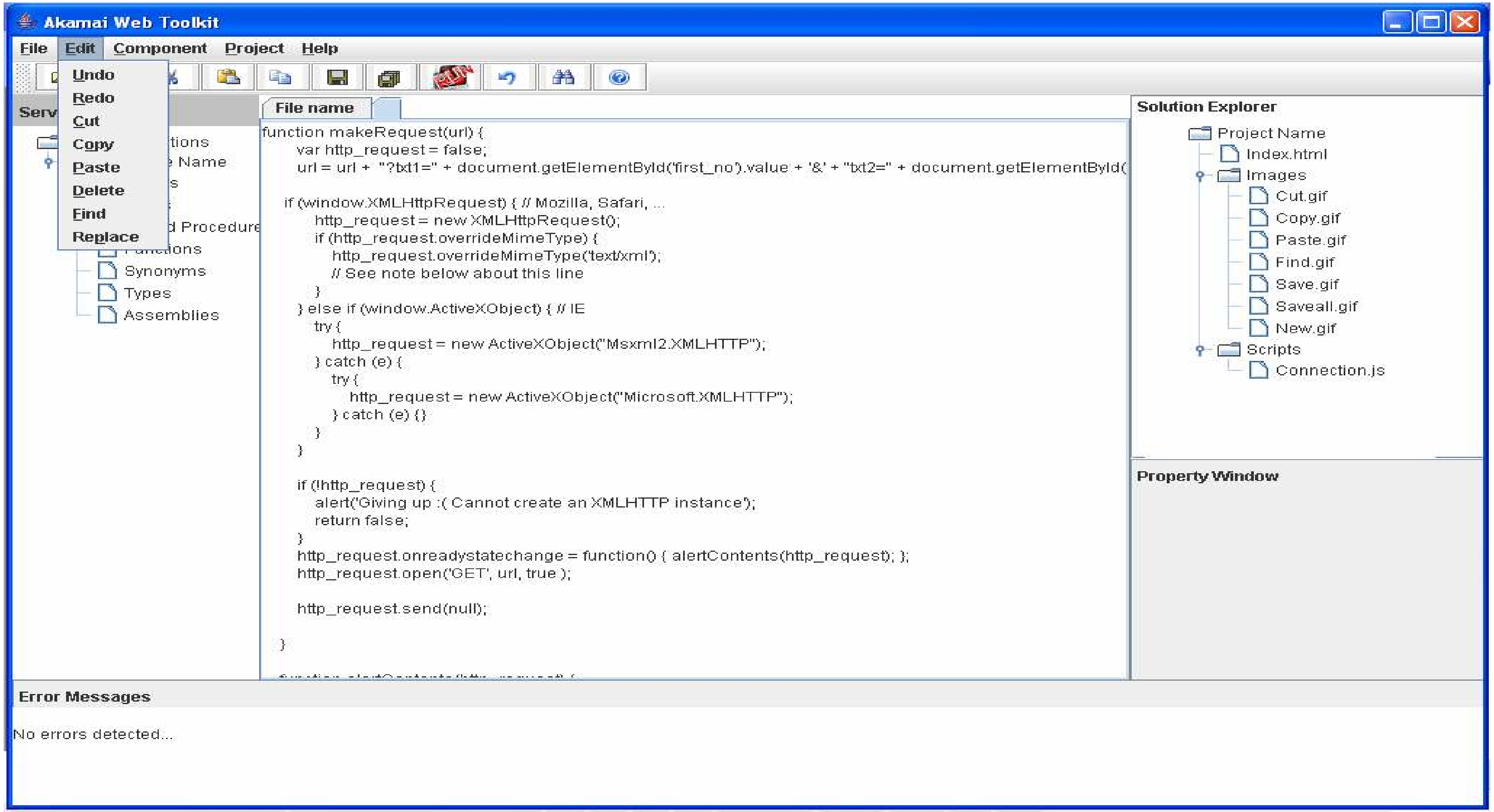
Above diagram shows the Akamai Web Toolkit Screenshot. In the menu bar we have five menus which are file, edit, component, project and help. In the toolbar you can see shortcuts for the program. In left side of our program users can see the database connections and in the right side we have the project directory and property window. In middle part of our program we have the text area for coding. Finally at the bottom users can see the error messages when compiled and run the code.



```
function makeRequest(url) {
      var http_request = false;
      url = url + "?txt1=" + document.getElementById('first_no').value + '&' + "txt2=" + document.getElementById(

    if (window.XMLHttpRequest) { // Mozilla, Safari, ...
         http_request = new XMLHttpRequest();
         if (http_request.overrideMimeType) {
            http_request.overrideMimeType('text/xml');
            // See note below about this line
         }
      } else if (window.ActiveXObject) { // IE
         try {
            http_request = new ActiveXObject("Msxml2.XMLHTTP");
         } catch (e) {
            try {
               http_request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
         }
      }

      if (!http_request) {
         alert('Giving up :( Cannot create an XMLHTTP instance');
         return false;
      }
      http_request.onreadystatechange = function() { alertContents(http_request); };
      http_request.open('GET', url, true);

      http_request.send(null);

   }
```

No errors detected...

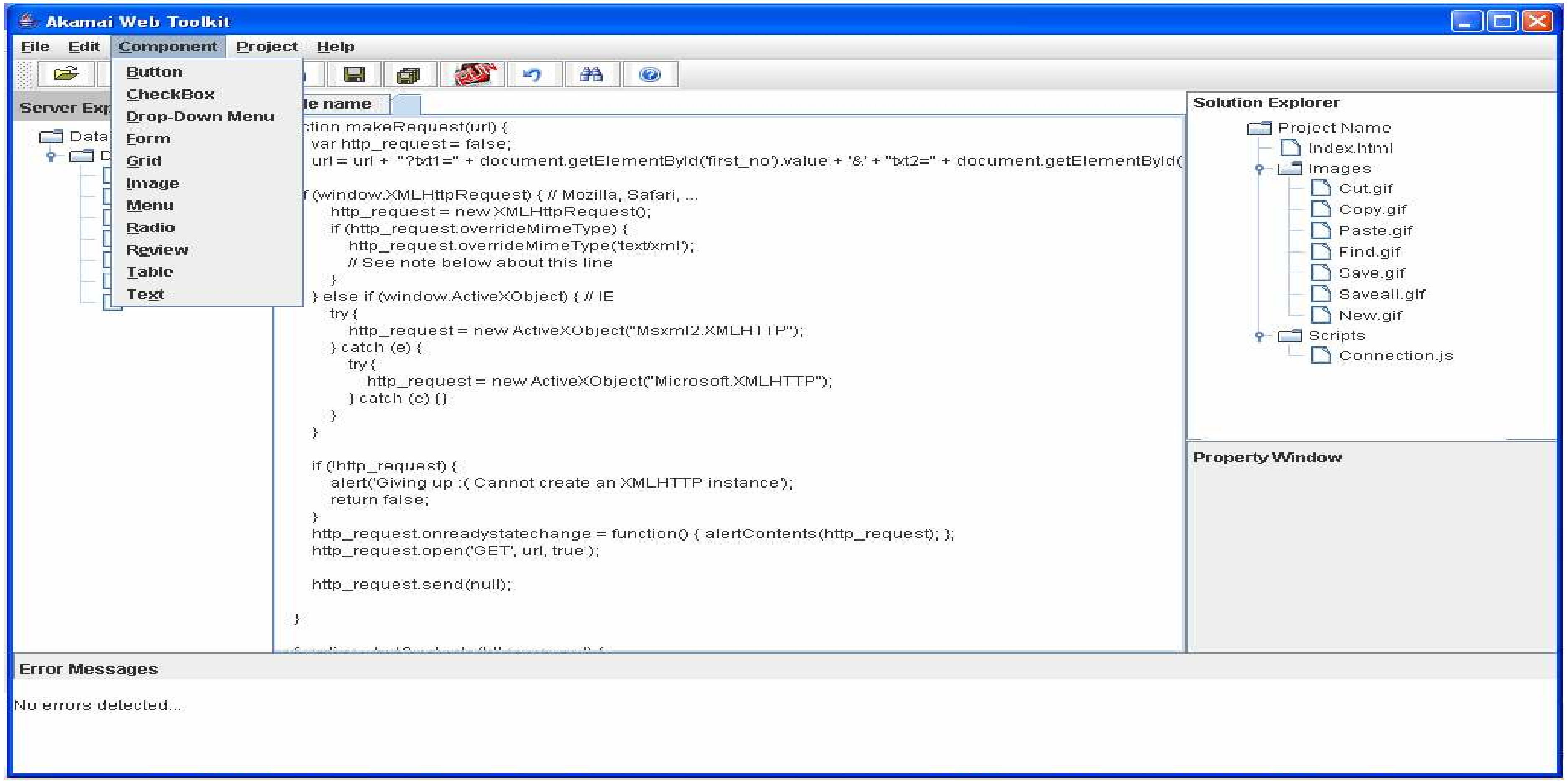
Above screen you can see menu items in file menu; these are open, save, add, close and exit. By using the open menu item you can be able to open a new or existed project. By save menu item you can save the project in selected folder. Add menu item is adds a new file. Close menu item closes the project and finally there is exit menu item, which exits from the program.



```
function makeRequest(url) {
    var http_request = false;
    url = url +  "?txt1=" + document.getElementById('first_no').value + '&' + "txt2=" + document.getElementById(

  if (window.XMLHttpRequest) { // Mozilla, Safari, ...
      http_request = new XMLHttpRequest();
      if (http_request.overrideMimeType) {
        http_request.overrideMimeType('text/xml');
        // See note below about this line
      }
    } else if (window.ActiveXObject) { // IE
      try {
        http_request = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (e) {
        try {
          http_request = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {}
      }
    }

    if (!http_request) {
      alert('Giving up :( Cannot create an XMLHTTP instance');
      return false;
    }
    http_request.onreadystatechange = function() { alertContents(http_request); };
    http_request.open('GET', url, true );

    http_request.send(null);

  }
```

No errors detected...

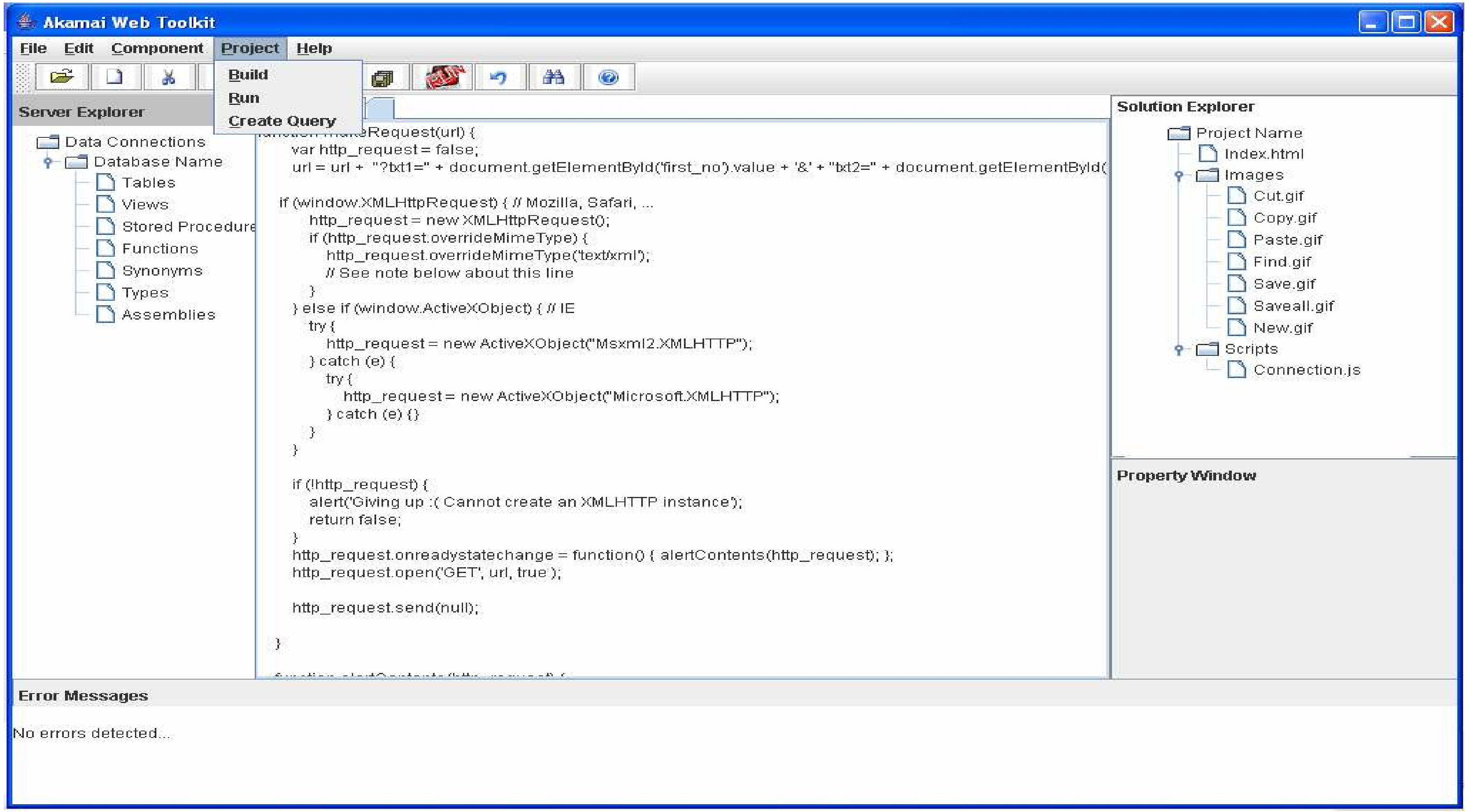
Above screen you can see menu items for edit menu. These are; undo, redo, cut, copy, paste, delete, find and replace. By these menu items you can do all the text edit operations.



```
ction makeRequest(url) {
    var http_request = false;
    url = url +  "?txt1=" + document.getElementById('first_no').value + '&' + "txt2=" + document.getElementById(

f (window.XMLHttpRequest) { // Mozilla, Safari, ...
        http_request = new XMLHttpRequest();
        if (http_request.overrideMimeType) {
            http_request.overrideMimeType('text/xml');
            // See note below about this line
        }
    } else if (window.ActiveXObject) { // IE
        try {
            http_request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                http_request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
        }
    }

    if (!http_request) {
        alert('Giving up :( Cannot create an XMLHTTP instance');
        return false;
    }
    http_request.onreadystatechange = function() { alertContents(http_request); };
    http_request.open('GET', url, true );

    http_request.send(null);

}
```

No errors detected...

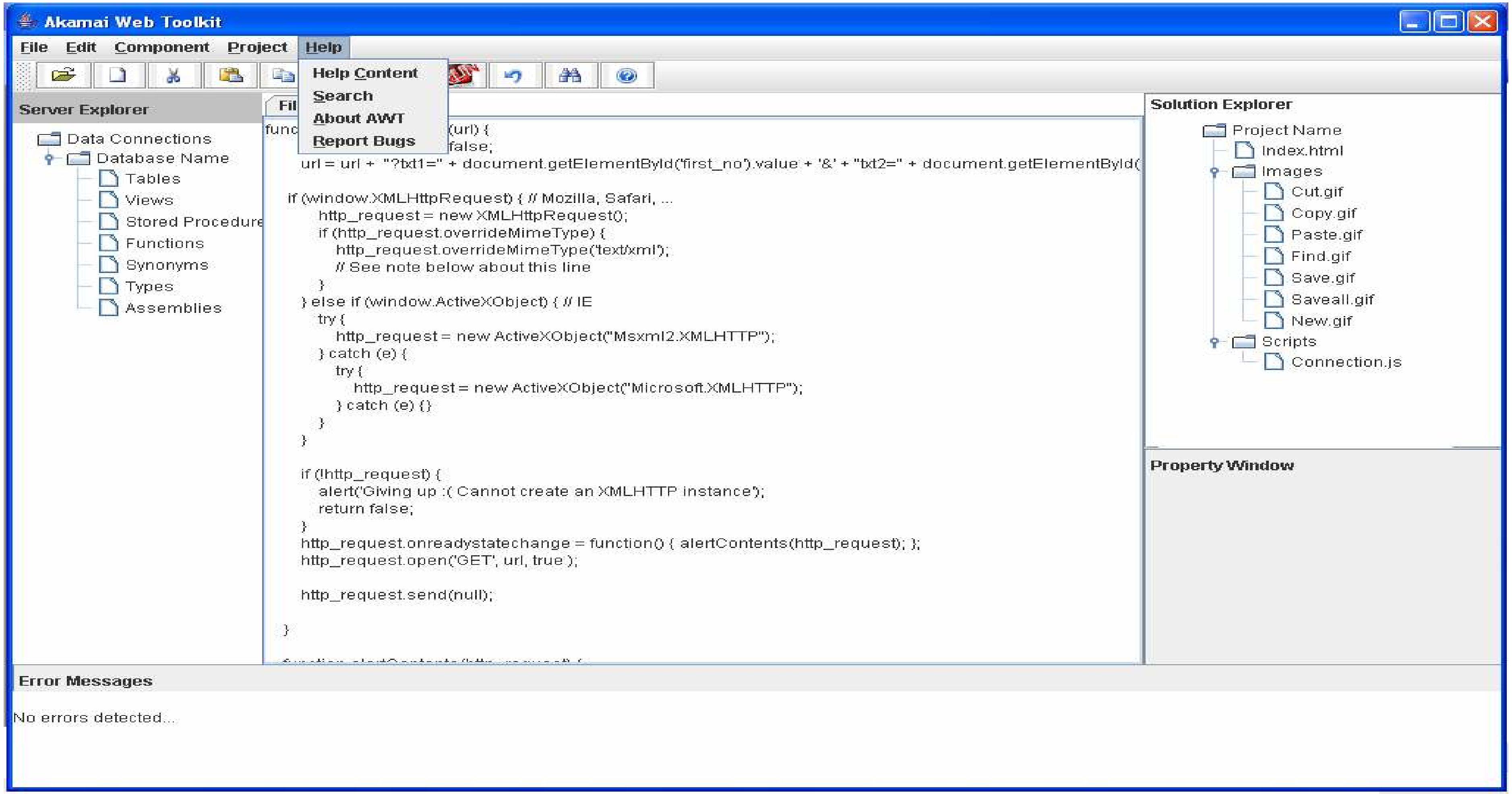
Above screen you can see the menu items for component menu. These are; button, check-box, drop-down menu, form, grid, image, menu, radio button, review, table and text. These are ready components that our users can add their code by using the menu items.



```
function makeRequest(url) {
    var http_request = false;
    url = url + "?txt1=" + document.getElementById('first_no').value + '&' + "txt2=" + document.getElementById(

  if (window.XMLHttpRequest) { // Mozilla, Safari, ...
      http_request = new XMLHttpRequest();
      if (http_request.overrideMimeType) {
        http_request.overrideMimeType('text/xml');
        // See note below about this line
      }
    } else if (window.ActiveXObject) { // IE
      try {
        http_request = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (e) {
        try {
          http_request = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {}
      }
    }

    if (!http_request) {
      alert('Giving up :( Cannot create an XMLHTTP instance');
      return false;
    }
    http_request.onreadystatechange = function() { alertContents(http_request); };
    http_request.open('GET', url, true);

    http_request.send(null);

  }

  function alertContents(http_request) {
```

No errors detected...

Above screen you can see the menu items for project menu. These are; build, run and create queries. User can build project in order to check for errors. If user clicks the run she/he can see the web browser. User can create query for getting desired information from table.



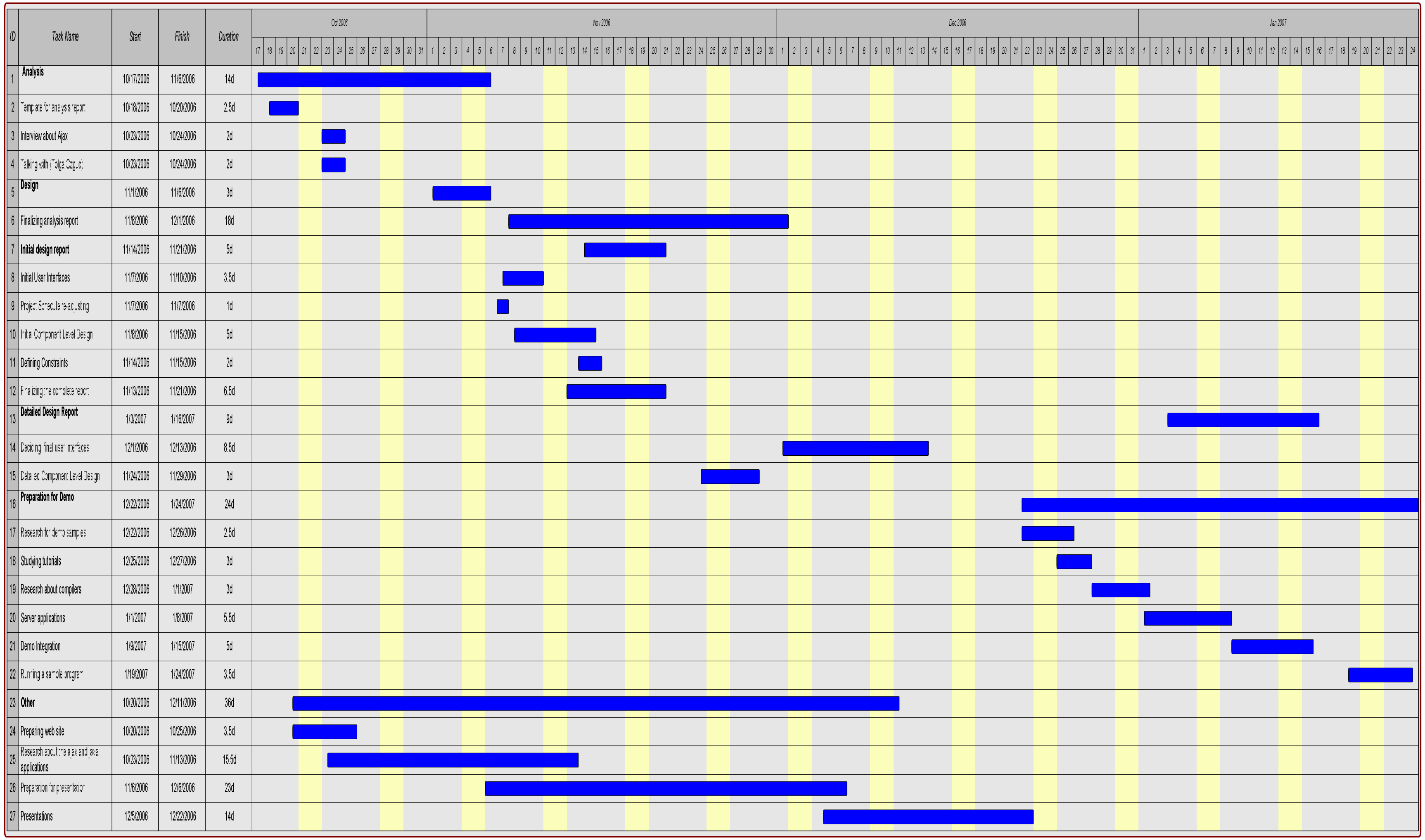
```
func                    (url) {
                        false;
      url = url + "?txt1=" + document.getElementById('first_no').value + '&' + "txt2=" + document.getElementById(

   if (window.XMLHttpRequest) { // Mozilla, Safari, ...
         http_request = new XMLHttpRequest();
         if (http_request.overrideMimeType) {
            http_request.overrideMimeType('text/xml');
            // See note below about this line
         }
      } else if (window.ActiveXObject) { // IE
         try {
            http_request = new ActiveXObject("Msxml2.XMLHTTP");
         } catch (e) {
            try {
               http_request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
         }
      }

      if (!http_request) {
         alert('Giving up :( Cannot create an XMLHTTP instance');
         return false;
      }
      http_request.onreadystatechange = function() { alertContents(http_request); };
      http_request.open('GET', url, true);

      http_request.send(null);

   }

   function alertContents(http_request) {
```

No errors detected...

Above screenshot is shows the help menu item. In this help menu it includes the help content, search, about AWT and report bugs. In help content user can get any information about the program. Search is used for finding the related contents in help content. About AWT is basically includes the versions and credits. Users can report bugs to us by using this menu item.

4. APPENDIX

A.



ID	Task Name	Start	Finish	Duration
1	**Analysis**	10/17/2006	11/6/2006	14d
2	Template for analysis report	10/18/2006	10/20/2006	2.5d
3	Interview about Ajax	10/23/2006	10/24/2006	2d
4	Talking with (Tolga Ozguc)	10/23/2006	10/24/2006	2d
5	**Design**	11/1/2006	11/6/2006	3d
6	Finalizing analysis report	11/8/2006	12/1/2006	18d
7	**Initial design report**	11/14/2006	11/21/2006	5d
8	Initial User Interfaces	11/7/2006	11/10/2006	3.5d
9	Project Schedule re-adjusting	11/7/2006	11/7/2006	1d
10	Initial Component Level Design	11/8/2006	11/15/2006	5d
11	Defining Constraints	11/14/2006	11/15/2006	2d
12	Finalizing the complete report	11/13/2006	11/21/2006	6.5d
13	**Detailed Design Report**	1/3/2007	1/16/2007	9d
14	Deciding final user interfaces	12/1/2006	12/13/2006	8.5d
15	Detailed Component Level Design	11/24/2006	11/29/2006	3d
16	**Preparation for Demo**	12/22/2006	1/24/2007	24d
17	Research for demo samples	12/22/2006	12/26/2006	2.5d
18	Studying tutorials	12/25/2006	12/27/2006	3d
19	Research about compilers	12/28/2006	1/1/2007	3d
20	Server applications	1/1/2007	1/8/2007	5.5d
21	Demo Integration	1/9/2007	1/15/2007	5d
22	Running a sample program	1/19/2007	1/24/2007	3.5d
23	**Other**	10/20/2006	12/11/2006	36d
24	Preparing web site	10/20/2006	10/25/2006	3.5d
25	Research about the ajax and java applications	10/23/2006	11/13/2006	15.5d
26	Preparation for presentation	11/6/2006	12/6/2006	23d
27	Presentations	12/5/2006	12/22/2006	14d