AJAX DEVELOPMENT ENVIRONMENT

TEST SPECIFICATION REPORT

1. INTRODUCTION 3
1.1 Goals And Objectives 3
1.2 Scope Of The Document 3
2. STATEMENT OF TESTING PLAN 3
2.1 Software for Testing 4
2.2 Testing Strategy 4
2.2.1 Unit Testing 4
2.2.2 Integration Testing 4
2.2.3 Validation Testing 5
2.3 Testing Tools and Environment 5
2.5 Test Schedule 5
3. MODULE TEST SPECIFICATIONS 5
3.1 Editor Module Test Specifications 5
3.2 Debugger Module Test Specifications 7
3.3 Explorer Module Test Specifications 7
3.3.1 File Explorer Module Test Specifications 7
3.3.2 Solution Explorer Module Test Specifications 7
3.4 Design Editor Module Test Specifications 8
3.5 Database Module Test Specifications 9
3.6 FTP Module Test Specifications 9
3.7 CVS Module Test Specifications 12

1. INTRODUCTION

This section gives a general overview of the test specification for an Integrated Development Environment for Ajax, kajax.

Our product **kajax** is simply an Integrated Development Environment for Ajax with the following main modules:

- Editor Module
- Debugger Module
- Explorer Module
- Design Editor Module
- Database Module
- FTP Module
- CVS Module

These modules are being implemented by one or two group members. Test specifications of each module are written by the implementer(s) of the corresponding module. Each module will be tested by the responsible member according to these test specifications.

1.1 Goals And Objectives

A sound product should work perfectly, doing the right thing at the right time. To do that the software has to go through a series of tests before its final release. Error free software is extremely difficult to achieve. Especially for software developed in a short time frame but high quality can be achieved with a detailed test specification. All (or at least most) of the test case will be listed, the development team will follow it step by step, item by item, to test all the necessary objects, data flows, limits, boundaries, and constraints of the software.

Karınca Teknoloji would like to have a test specification to counter any difficulties that may impact the development and the future performance of the software.

1.2 Scope Of The Document

An overall plan for integration of the software and a description of specific tests are documented in this section. Below are the different kinds of tests that we will take to ensure the quality of the software.

- Unit Testing: All the modules will be tested individually performing black box testing methods.
- Integration Testing: Related modules are integrated and tested in order and all the system is tested.
- Validation Testing: We will test software as a whole, so all the units of the software will be included.

2. STATEMENT OF TESTING PLAN

The product has to be bug free. The testing procedure and strategy that will be followed is the following:

2.1 Software for Testing

The functionalities of **kajax** modules are described entirely in the final design report. Mainly, software to be tested is our Integrated Development Environment **kajax** i.e. its modules.

2.2 Testing Strategy

The methods for our testing strategy are the followings:

2.2.1 Unit Testing

Unit testing is used for testing separate modules of **kajax.** White box testing will be done for each module of the software individually. Components will be tested by passing related data to each of them. Expected results will be monitored for correctness and errors. Entrance and exist conditions of data will be observed. As a result of these processes, Testers will be sure about that each module is working well. Unit testing specifications will be presented in section 3.

2.2.2 Integration Testing

When we are confident that all individual modules are working well, we will begin the integration test of the modules. Namely, after unit testing, we are going to make some commitment about this phase of testing.

In integration test we are using bottom up modeling. The reason why we determined the bottom up testing is that it is easy to combine the little pieces. We divided our system into the several modules. They are implemented independently by each member.

Moreover, we are not able to do integration test from beginning of the implementation to the end of the implementation. We determined some milestones for each module to be finished.

Only when they are finished we can integrate them and test them in this phase. However in some cases, in order to test some modules we have to use the other modules.

In the following, we listed the modules which are interacted to other modules. From the table, it is easily seen that which module will be integrated to which one.

	First Module	**Second Module**	**Description**
1	Solution Explorer Module	Editor Module	Second module gets the file info from the first one.
2	Database Module	Editor Module	First module sends the connection file to second module.
3	Design Module	Editor Module	First module sends the necessary codes to second module
4	FTP Module	Explorer Module	Second module uses the output of the first module.
5	CVS Module	Explorer Module	Second module uses the output of the first module.
6	Debugger Module	Editor Module	First module uses the output of the second module.

2.2.3 Validation Testing

The software requirement report will be controlled by the product to see whether there is a conflict between them. Black box testing is performed to see if the software is completed. The modules will be tested all together. Test Inputs will be provided to the software and the outputs will be compared with the expected ones, so that validation of the product will be checked. If an error occurs, this will be reported to the implementer of the related module.

2.3 Testing Tools and Environment

Testing tools are same as our development environment. We will use our computers for testing just as for developing.

- **Netbeans IDE 5.5 with JAVA JDK 1.6**

2.5 Test Schedule

Test plan delivery	06.05.2007
Unit Testing	15.05.2007 – 18.05.2007
Integration Testing	19.05.2007 – 21.05.2007
Validation Testing	22.05.2007 – 23.05.2007
Correction	23.05.2007 – 01. 06.2007
Final Test	02.06.2007 – 09.06.2007

3. MODULE TEST SPECIFICATIONS

3.1 Editor Module Test Specifications

Main test points in Editor Module Test are the followings:

- Opening, closing and saving files.
- Searching and replacing patterns in the files.
- Jumping to a line.
- Toggling line number panel.
- Toggling line wrapping property.
- Colorizing codes according to their syntax.
- Assisting user while editing files by suggesting completions.
- Automatic matching pairs like braces, quotes, etc.
- Making automatic indentations to help user.
- Making undo/redos.

Testing steps:

Test Case#	Description	Test Steps	Expected Results
1	Opening a file	User selects a file to open. Then presses open button.	If the file is not opened already then it is opened in a new tab.
2	Closing a file	User selects an editor tab to close. Then presses close button.	If there is no change in file the tab is simply closed.
3	Saving a file	User selects an editor tab to save its content. Then presses save button.	The file is saved without prompt if it was an existing one, else user prompted to enter file name.
4	Searching for a text	User enters text to be searched. Makes settings for the search. Then presses find button.	Found pattern is highlighted. If no pattern remains user is warned up.
5	Replacing a text	User enters text to be searched and text to be replaced. Makes settings for the replacement. Then presses replace(or replace all) button.	Found patterns are replaced. End of replacement is notified to the user.
6	Jumping to a line	User enters the line number to go. Then presses go button.	Cursor is directed to the specified line. If it passes the end of line, cursor is positioned to the last line. For non-numeral inputs user is warned.
7	Toggling line number panel	User presses toggle line number panel button.	Line number panel is shown or hidden according to its previous state.
8	Toggling line wrapping	User presses toggle line wrapping button.	Lines that are beyond the view port are wrapped and line numbers are arranged accordingly.
9	Syntax Highlighting	User make changes on a document with a supported type.	Syntax is highlighted according to the type of the document.
10	Automatic indentation	User presses enter key.	Cursor is indented to the last tab character of the previous line.
11	Code Assistance	User presses ctrl+space. Then selects an entry among the items. Then presses enter.	A list of possible code fragment or words shown to the user. And the selected one is pasted to the editor.
12	Pair matching	User enters quote, parenthesis, or braces.	Pairs are automatically matched and pasted into the editor.

13	Undoing/Redoing	User presses ctrl+z(ctrl+y).	Last change is taken back. (Or redone.)

3.2 Debugger Module Test Specifications

Test Case#	Description	Test Steps	Expected Results
1	Execute/Stop Command	Whether user can stop or restart the running process of the project	Execution control should be provided.
2	Add/Remove Breakpoint	Whether user can add or remove breakpoints from the editor	Execution should be stopped when it comes to the determined breakpoints.
3	Step in/ step over/ step out	Whether user can select the type of execution process	Execution process should carry on using the continuing commands of user
4	Add/Remove watch variable	Whether the user can add a watch variable to see the values.	Values of selected variables should appear accordingly.

3.3 Explorer Module Test Specifications

3.3.1 File Explorer Module Test Specifications

Test Case#	Description	Test Steps	Expected Results
1	Creating a file	Whether user can create a HTML, Javascript,XML,CSS files	New file is created and seen in the GUI.
2	Deleting a file	Whether user can delete a HTML, Javascript,XML,CSS files	Deleted file should be removed from the kajax gui and persistent storage
3	Monitoring Hard Drive	Whether user can display all directories and XML,Javascript,CSS,HTML files	All these files should be displayed.

3.3.2 Solution Explorer Module Test Specifications

Test Case#	Description	Test Steps	Expected Results
1	Creating a project/file	Whether user can create a valid project/file or not using project/file wizard.	New project/file is created.
2	Deleting a project/file	Whether user can delete a valid project/file from persistent storage	Deleted file or project should be removed from the kajax gui and persistent storage
3	Closing a project	Whether user can remove the project from the kajax gui.	Deleted file or project should be removed from the kajax gui.
4	Setting main	Whether the user can adjust the	When the user presses the run

	project	desired project.	button main project adjusted by the user should run.
5	Active Projects	Whether all the opened projects can be monitored by the user	All active projects should be seen in the solution explorer panel.
6	Opening Project	Whether user can browse local files and recognize the kajax projects.	Kajax projects should be displayed to the user during browsing.

3.4 Design Editor Module Test Specifications

Main test points in Design Editor Module Test are the followings:

- Adding objects to design editor
- Manipulating properties of the objects
- Manipulating events of the objects
- Following text-to-design and design-to-text reflections.
- Previewing

Testing Steps:

Test Case #	Description	Test Method	Expected Result
1	Adding components from the insert menu	User adds desired components from the insert menu.	Components are put in editor panel; related properties and events are displayed.
2	Manipulating Properties	Properties of the components are changed.	Results of changes are reflected.
3	Manipulating Events	Events are selected from combo boxes.	Components react to events accordingly.
4	Generating Code	Freely play with design editor.	Corresponding code is generated accordingly.
5	Updating Design	Change code in text editor.	Desing editor is updated accordingly.
6	Previewing	Preview the designed page in embedded browser	Page be displayed accurately.

3.5 Database Module Test Specifications

Main test points in Database Module Test are the followings:

- Setting up a connection with database server.
- Querying database and viewing results
- Making .php or .asp files to get data from database server.

Testing steps:

Test Case#	Description	Test Steps	Expected Results
1	Setting up a connection with database server	User Enters Data Source, Host Address, Port Number, Database, User Name and Password. User presses OK button.	Connection is established. User is directed to the main GUI and tables will be seen with tree view below the file explorer.(Database Panel)
2	Querying database and viewing results	User selects the "Make Query" from the menu. From the "Database window" user writes queries and presses "RUN" button.	The results of the query will be displayed in the result table.
3	Making .php or .asp files to get data from database server.	User double clicks the table name in the database panel.(In Main Menu)	A new file will be opened in the editor which includes necessary codes to retrieve the selected table data either in .php or .asp.

3.6 FTP Module Test Specifications

Main test points in FTP Module Test are the followings:

- Setting up a connection with a remote host
- Making FTP site configurations
- Creating a connection between the FTP side and local side
- Transferring files from FTP side to local side
- Transferring files from local side to FTP side
- Editing files at the remote host
- Synchronizing between different FTP sites
- Disconnecting from a FTP connection
- File Explorer functions for both Local Side and FTP site which are browsing files, filtering files by their type and ordering the files.
- Deleting, Copying, Renaming files from both local and ftp side and making new directories.
- Entering console commands.

Testing steps:

Test Case#	Description	Test Steps	Expected Results
1	Setting up a remote host connection	User Enters User Name, Password, Port Number and Host Name. User presses login button.	Connection is established. User is directed to the kajaxFTP main GUI. File Explorer and Console is active.
2		User chooses a previously made FTP site configuration. Then, user presses connect button.	Connection is established. User is directed to the kajaxFTP main GUI. File Explorer and Console is active.
3	New Login	User presses new login button.	Disconnected from the current connection. Returning to the Connection Wizard.
4	Log Out	User presses to the log out button.	User disconnects from FTP side. FTP side file browser is disabled.
5	Synchronizing between different FTP sites	User makes this by New Login action.	User synchronizes between different FTP sites
6	Browsing files in local side	Users browse local files by using local side browse combo box.	Local files are updated in local file table (whose columns are file name, property and last modified) according to the user selections.
7	Browsing files in ftp side	Users browse ftp files by using ftp side browse combo box	Ftp files are updated in ftp file table (whose columns are file name, property and last modified) according to the user selections.
8	Local side file type filtering	In local side file type combo box, User selects one of the following options: Text Files, Html Files, Script Files, Java Class Files, Image Files, Source Files, Database Query Files, and Batch Files.	Local files in the local file table are filtered according to user selection.
9	Ftp side file type filtering	In ftp side file type combo box, User selects one of the following options: Text Files, Html Files, Script Files, Java Class Files, Image Files, Source Files, Database Query Files, and Batch	Ftp files in the ftp file table are filtered according to the user selection.

		Files.	
10	Copying a file from FTP side to local side	User selects a file from FTP side by clicking the file in the FTP side file table. Then presses copy button.	The file is transferred to the current location of local file browser.
11	Copying a file from local side to FTP side	User selects a file from local side by clicking the file in the local side file table. Then presses copy button.	The file is transferred to the current location of the ftp file browser.
12	Deleting a file from local side	User selects a file from local side by left clicking the local file table. Then press delete button.	The file is deleted from local side.
13	Deleting a file from FTP side	User selects a file from FTP side by clicking the ftp file table. Then presses delete button.	The file is deleted from FTP side.
14	Deleting a directory from local side.	User selects the directory by clicking local side. Then presses delete button.	The selected directory is deleted from local side.
15	Deleting a directory from FTP side.	User selects the directory by clicking ftp side file table. Then presses delete button.	The selected directory is deleted from FTP side.
16	Renaming a file in local side	User selects the file to be renamed in local file table. Then presses rename button. Enters the new name.	The selected file is renamed in local side.
17	Renaming a file in FTP side	User selects the file to be renamed in FTP file table. Then presses rename button. Enters the new name of the file.	The selected file is renamed in FTP side.
18	Renaming a directory in local side.	User selects the directory to be renamed by clicking the local file table. Then presses rename button. Enters the new name of the directory.	The selected directory is renamed in local side.
19	Renaming a directory in ftp side	User selects the directory to be renamed by clicking the FTP file table. Then presses rename button. Enters the new name of the directory.	The selected directory is renamed in FTP side.
20	Making a new directory in local side	User selects the directory location by browsing local side browser. Then presses new directory button. Enters a name for directory.	A new directory is made in local side.
21	Making a new directory in FTP side	User selects the directory location by browsing FTP side browser. Then presses new directory button. Enters a name for directory.	A new directory is made in FTP side.

22	Console Testing	User makes an FTP action. For example, deletes a file from local side.	Console text area is updated with the related FTP commands for each specific action.
23	Commanding an FTP command	User enters an FTP command from console command line.	Command is done and console text area is updated with the related command action results.
24	Making an FTP site configuration	User enters User Name, Password, Port Number and Host Address and presses Test Button and saves it with a name by pressing save button.	New site configuration is listed in configuration list.
25	Deleting an FTP site configuration.	User selects the FTP configuration to be deleted and presses delete button.	Site Configuration is deleted and configuration list is updated.

3.7 CVS Module Test Specifications

Main test points in CVS Module Test are the followings:

- Setting up a connection with a remote CVS host
- Making CVS server configurations
- Invoking CVS Commands i.e. Status, Remove, Log, Diff, Tag, Update, Commit, Check Out, Add.
- Handling CVS Events i.e. File Added, File Info, File Removed, File Updated, Take Message.
- Disconnecting from an established CVS host.

Testing Steps:

Test Case#	**Description**	**Test Steps**	**Expected Results**
1	Setting up a connection with a remote host	User enters Protocol, Protocol Parameters, Server Name, Port, Repository folder, User Name and Module to be connected.	Connection is established with the remote CVS host.
2		User selects a previously defined CVS configuration. Then presses login button.	Connection is established with the remote CVS host.
3	Making CVS server configurations	User enters Protocol, Protocol Parameters, Server Name, Port, Repository folder, User Name and Module to be connected for a specific CVS server configuration. Then user presses test button to test it. User saves it by pressing	CVS configurations list is updated.

		save button.	
4	CVS Commands	User makes an action for invoking CVS commands which are status, remove, log, dif, tag, update, commit, and checkout, add.	Related output of the command is expected. For example if a remove command is invoked, the files and directories to be removed has to be removed from CVS server.
5	CVS Events	CVS client has to be informed by the changes. These changes may be File Added, File Info, File Removed, File Updated, and Take Message.	Related output of the CVS event is expected. For example, when a File Added event occurs, CVS clients have to be informed by this event.
6	Disconnecting from a CVS host	User presses the disconnect button.	Disconnected from the remote CVS host.