MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

SENIOR PROJECT

SPRING 2008

TESTING SPECIFICATION PLAN

LOGICIEL

ONUR AK 1394576 **SADETTİN ŞEN 1395540**

MASHAR TEKİN 1395565 **ŞERİF ÇETİNER 1394832**

TURKUAZ

UMUT EROĞUL

INDEX

1. INTRODUCTION

This section gives a general overview of the test specification for our project Report Separator a text miner for free text radiology reports.

1.1. Goals and Objectives

The Report Separator project has several modules including user interface, search, text mining and database modules. We are assured that processing and duties of each module should be verified so that all of these modules work in harmony and correctly when they are integrated. As a result, we give great importance to testing our project in order to obtain a bug free, and logically correct product with high performance.

1.2. Scope of Document

The purpose of this document is to describe the testing process of the project Report Separator. In fact, while developing our project up to now, we did testing for each module, so we will explain about the testing process that took place since the beginning of the project until now.

1.3. Statement of Testing Plan Scope

Testing process of the project Report Separator includes unit testing, integration testing, performance testing and alpha- beta testing.

Unit testing: For the modules user interface, search, text mining and database.
Integration testing: This is the most important part of our testing because getting a whole program that works correctly is the aim of this project.
Performance testing: In this part we tested the accuracy of results and the speed of program.
Alpha-Beta testing: Whenever a new feature was added to the project someone in our group made alpha testing so we could be sure that the added parts work correctly. After all of the parts were added all group members made beta testing for being sure that our project was bug-free.

1.4. Major Constraints

We had several different constraints in our testing plan. For our project most important constraint was time since we had to finish the final package on time while we were testing our software. Another constraint of our testing plan was the number of staff. We had 4 group members and it made difficult to do all tests in time. Since our project could work on a standard computer we had no hardware constraints.

1.4.1. Time

Since our project should have been finished due to 15 June, time was the main constraint for us. We had a schedule for the testing phase, therefore the deadlines of each specific work were known beforehand. If we managed to obey the deadlines according to schedule, our project was released after proper testing and in time.

1.4.2. Staff

Number of staff was a major constraint for our testing plan since there were only four people, who had also other responsibilities in our project, as a staff.

2. TESTING PLANS AND STRATEGIES

In our testing plan we had unit testing, integration testing, performance testing and alpha-beta testing.

2.1. Unit Testing

While developing our project by proceeding module by module, we were making sure that every module was processing correctly so that when these modules were combined, we could be sure that there were no problems in the basement. We believed that the modules could work in harmony more easily by proceeding in this bottom up manner. We tested search, text mining and user interface modules manually. On the other hand database module tested with the help of the program that we had coded and with the help of a database management

program pgAdmin III. This program has a GUI for managing postgreSQL database.

2.1.1. User Interface Module Testing

Tests of user interface functionalities. All functionalities of user interface were also tested whenever they were added in the scope of alpha testing. In our user interface testing major functionalities that we tested are:

1. 'Dosya' tab: Selecting a file and displaying the content of file.

2. 'Düzelt' tab: Displaying unknown words together with suggestion for each word. Entering correct form of unknown words and tagging these words.

3. 'Etiketle' tab: Displaying sentences of reports, selecting sentences with the help of buttons 'Önceki' and 'Sonraki' and displaying the tagged words of selected sentences, selecting a word from these sentences by left clicking to the word and choosing a label for tagging this word.

4. 'Sonuç' tab: Displaying name of file and the results of text mining process.

5. 'Arama' tab: Supplying four different choices for search in four new tabs.

All other user interface functionalities were also tested here only the main parts were given.

2.1.2. Search Module Testing

Although testing of this module was relatively easy, for accomplishing this part we needed database module since all reports were stored in database. In search module we supplied four different type of search and each were tested individually. These parts were 'search for doctor name', 'search for similar reports', 'general search' and 'detailed search'. In each part we entered needed information for search and manually looked the reports, which names were given in the search result, whether they actually included the given information or not

2.1.3. Text Mining Module Testing

This part was tested twice. In first one we tagged some words with labels such as ‘yer’, ‘hastalık’ and ‘zaman’. After this tagging operation we tested some sentences that contained tagged words and determined whether our tagging process worked correctly or not. Since the accuracy of the results of text mining module shows the accuracy of our project we tested this module again after a training set was tagged. We checked results for seeing whether our method works correctly or not.

2.1.4. Database Module Testing

For testing database module we used a program coded by us that inserts all reports in a directory to database. This program also inserts each word and sentence to related tables of database. These tables are used for searching and tagging. Another way that we test database module was GUI functionalities of our project. We manually selected reports and inserted them to database. We use pgAdmin III which is software with GUI used for managing postgreSQL databases. With the help of this software we could see all tables in database and elements of tables easily.

2.2. Integration Testing

When we were confident that the all individual modules were working well, we began the integration test of the modules. In integration test we used bottom up modeling. The reason why we determined the bottom up testing was that it was easy to combine the little pieces. Moreover, we were 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 were finished we could integrate them and test them in this phase. However in some cases, in order to test some modules we had to use the other modules. Especially search module testing could not be tested without database module.

In the following, we listed the modules which were interacted to other modules. From the list, it is easily seen that which module needed to be integrated to which one

1. Search module – Database module

2. Text Mining module – Database module
3. User Interface module – Search module
4. User Interface module – Text Mining module

All these modules were tested after integrations of them.

2.3. Higher Order Testing

As at the end of this project we had to come with a full software package and we had to release this software to costumers, it was required to apply higher level test to our software package. We separated our high level tests as, performance and alpha-beta tests.

2.3.1. Performance Test

Since accuracy and time were the most important performance measure in our software we measured time for analyzing a radiology report and the accuracy of each report. Measuring time was relatively easy and we tested lots of different reports. For this testing we chose a report that was not currently in database and analyzed it. For accuracy test we manually checked results whether they were correct or not. We looked correctness of the ‘name of illness’, ‘area of illness’, ‘state of illness’, ‘dimension of illness’, ‘time’ and ‘suggestion’. Some of these were available in all reports while some of were not. If one was not in the report we neglected that part while calculating accuracy.

2.3.2. Alpha-Beta Testing

Every time we added a new feature to our software package, we tested our application and looked the results. Moreover, we tested the reliability and consistency with other features of the new feature. Consequently, our alpha testing continued during the semester. However, as we intended to come up with a reliable software package we had to do a beta test, to see whether we had any bug or inconsistency which had not recognized before. For this, all members of our group Logiciel tested the project’s different parts. While some of us testing search module in detail the others tested the database modules.

3. TESTING TOOLS AND ENVIRONMENTS

1. Net Beans IDE 6.0: For Java code developing and debugging
2.. pgAdmin III: For visualizing and managing database

4. STAFFING

The staff of our project was mainly responsible for the parts showed below:

Mashar Tekin: Testing text mining modules, Testing user interface
Onur Ak: Testing database modules, Testing user interface
Sadettin Şen: Testing database modules, Testing search modules
Şerif Çetiner: Testing text mining modules, Testing search modules

5. TESTING SCHEDULE

The following is the schedule for the testing plan that was presented in this report:

Test Plan Delivery: (Deadline) 23.05.2008
Unit Test and Integration Tests: (Start) 23.05.2008 - (Deadline) 01.06.2008
Performance Test: (Start) 01.06.2008 - (Deadline) 06.06.2008
Alpha and Beta Tests: (Start) 06.06.2008 - (Deadline) 13.06.2006
Results (i.e. Bugs) Tracing and Correction: (Deadline) 15.06.2006