

Online Adaptive Testing System (OATS)
The project documents for Agent based software engineering course

**Student Names: Mohammad Khabbazineh Tajer,
Keyvan Khosrovian Kermani**

Instructor: Dr. Behrouz Homayoun Far

Contents

1) System Specification	3
1.1) Business Case.....	3
1.2) System description	4
1.3) Assumptions.....	5
1.4) Requirements.....	7
1.5) Wish list.....	7
2) Conceptualization.....	8
2.1) Use cases.....	8
2.2) Actors.....	8
3) Agent Modeling.....	13
4) Task Modeling.....	21
5) System Architecture.....	22
6) Data Specification.....	23
7) Communication specification.....	26

1) System Specification

This document is produced to support the Analysis and design phases for an online testing system. In this part of document we will talk about the system specification. First, we will explain about the business case after that we will describe our system mention assumptions and requirements and at last we will include our wish list.

1.1) Business Case

Current testing systems in most schools and universities are still paper based systems. Paper based tests usually cause lots of difficulties and limitations for both instructors and students. Some of the difficulties for students taking current test styles are as follows:

- All of the students must be present at the same time at the same place. Sometimes the test schedule for students gets so tight that makes it really difficult for them to move to a certain place or to take exams at a pre specified time.
- Sometimes students have several tests in a row and the testing system does not give them the flexibility to schedule their tests themselves.
- In most cases students expect to receive the result of their test as soon as possible but paper based testing system is dependent on the instructors about this issue and sometimes some instructors are not able to prepare the results in a timely manner.
- There are lots of problems about the test reports even in case of using computers to mark the answer sheets for paper based tests. In most cases, test reports do not contain details about the given answer and correct answers for students.
- Computer based tests are also being used for some special courses in some special institutes but there are still limitations even with computer based tests. In most cases, students must go to a certain place to take the test.

And there are also lots of difficulties for instructors giving paper based tests:

- Instructors usually have a bank of questions and each time they have to search and select the question from their own question bank.
- Sometimes, they have to design several versions for the same exam to prevent students from cheating.
- For multi choice questions, some students try to apply specific patterns to answer the questions and some of them are really successful in receiving good marks without any studying!
- Some times it is better to design tests for students based on their skills. We want to make the test more difficult for students who have had a better performance while keeping the measure for final score the same for all students. It is really difficult and time consuming to do this and it also could be offensive for those students who have not had a good performance.
- Marking answer sheets is a very time consuming activity itself. And there are also lots of objections to the scores from students because of the poor report of score and test performance.

It is possible to avoid most of these difficulties applying an online testing system.

- We can offer several test sites which are connected to the internet and give students the flexibility to select among several locations for their test.
- It is also possible to let students schedule their tests before taking it to take the test in a suitable place at a suitable time.
- We can calculate students' score and show them the test report right after they finish their test. In this way, students will be informed about the result of their test and know the correct answer all of the questions and the number of objections about the test will be decreased.
- Instructors need not worry about the design of the test. We can assign the responsibility of designing the test to our online system.
- We can design adaptive tests which means the difficulty of test will be increased or decreased during the test based on the user performance dynamically.
- We can check the pattern of given answers to reduce the possibility of answering questions by chance.

Why do we need a multi agent system for this purpose?

We have already described the benefits of having an online test service. Now we have to explain why we need to make a multi agent system for this purpose. As we mentioned before, we need an online system which gives the students the flexibility of scheduling their test and we also want to create adaptive tests. Also, we need to produce our reports based of the difficulty of test. We can not apply the same scoring policy for all of the students. All of these attributes clearly show that in some cases we need to have intellectual behavior in our system and also we need to coordinate lots of issues in our system in order to provide a successful testing system. There are many possible ways to implement such kind of system. One way which seems to be suitable for this purpose is to implement a Multi agent online testing system. We can assign roles of people who are really taking the test in paper based style to our agents. We can have intelligent agents who are responsible for creating adaptive tests and are in charge of creating the report for test. Because the main purpose of our system is to simulate the existing human interactions with students such as scheduling, designing the sets with respect to student skills and also proper reports the selection of multi agents systems for the development of our system seems to be a reasonable choice.

1.2) System description

For implementing online testing systems we introduce our system named "Online Adaptive Testing System" (OATS). This system is designed to assist students with scheduling their test and also provides services for test registration. It has facilities for online payment for the test. To handle the payment we use online web services and use related protocols. Before registering for the test student should schedule it. The student provides the system with requested period of date and time and also the type of test. System tries to provide the possible existing choices based on the capacity of each test. If there is no possible test, system returns a message and asks the user to reschedule the test.

After scheduling the user has to perform the payment and register for the test. For this purpose, user provides the system with online payment options and system tries to perform the payment through the web services. If the payment is successful the transaction id is stored in the database for future reference and user is registered for the test. Then, system provides user with a username and password and also a test registration number which are valid for a certain period of time equal to the duration of test beginning from the start time of test for the registered test.

System is responsible for authentication before the test starts. After authentication the test starts and continues in an adaptive way as described before. When the test is finished user requests for report and system calculates user's mark dynamically based on the difficulty level of answered questions and provides the report for the user. Another important task which is performed by our system is to update initial the difficulty level of questions in database at certain points of time. This updating activity is performed based on the number of correct and incorrect answers to the question and also the detected skill of student who has given that answer.

In our testing system, Questions are multi choice questions so as mentioned before, the probability of answers by chance exists. The system also detects the pattern of given answers and decides about the position of next correct choice. In order to perform mentioned tasks the system uses online web services and also is connected to question database. We should also provide a suitable user interface for the students while they are taking the test. It is important to show each question in the best possible way. We also need user interface for registration and also for showing the reports to students.

Figure 1 shows an overall view of our system.

1.3) Assumptions

There are several assumptions for the implementation of this system.

- We have a complete database of pre inserted questions in test database.
- There is an available web service on internet for online payment and this service is provided with most of the well known banks. We can use the web service with the same WSDL protocol and syntax for all of the banks
- The difficulty level of test will be changed while the user is taking the test according to the user performance
- The difficulty level of each question is also subject to change during the time
- System keeps the record of students after they finish their test for the future usage to adjust the difficulty level of questions
- Each test takes place in a test site and the system is only responsible about the username and password for user authentication.
- All of the questions are stored in a centralized database
- For the scheduling task, user provides the system with the desired period of time and the desired place for taking the sites. User selects among the existing available test places.
- User has to schedule the test before registering for it

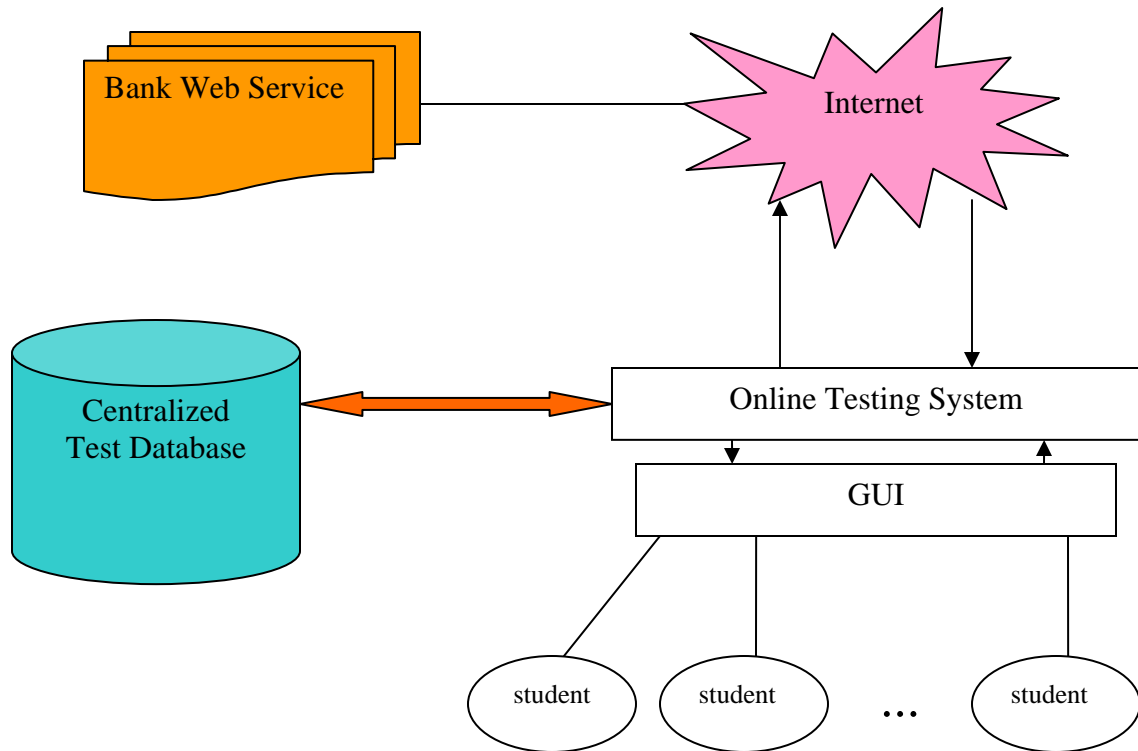


Figure1. Overall view of OATS, required resources and interactions

- All of the tests in our system require payment
- User can register only for one test each time
- User has to use online payment options to pay for the test
- Username, password and test registration number are needed for authentication. Without this information, the user is not allowed to start the test.
- Each test has a test type which determines the course from which this test will be taken
- Each test has a limited capacity, after the capacity is full, that test section won't be available anymore. By test section, we mean a certain test type which takes place in a certain duration of time in a certain place
- All of the questions for our testing system are multi choice questions
- After the test is finished, the user's report will be prepared instantly
- Data entry for new tests and questions is done through another test management system which is responsible for updating the database and it is not an online system necessarily

1.4) Requirements

The list of requirements for OATS is as follows:

- OATS has the responsibility of providing the student with all of the possible choices for scheduling the test
- The system is responsible to process the payment and ensure the security of payment
- System must provide the user with a valid username, password and test registration number for the required test section
- The system is responsible for change the difficulty of test dynamically while the user is taking the test
- The system must evaluate the user based on the difficulty level of test
- The system is responsible for changing the difficulty of questions gradually
- The system must check for any possible patterns in the correct answers given by the user to design the position of correct answers in a way that does not match the detected pattern
- System is responsible for providing the user with the test report right after the test finishes

1.5) Wish List (Not Implemented)

- The user should have the ability to reschedule the test
- The system must be capable of accepting several payment methods
- The system should be capable of canceling a test
- User should be able to register for several tests with a single payment
- Essay questions must be considered for the system
- The system should be capable of working with distributed question databases from different vendors

2) Conceptualization

In the conceptualization phase the main effort is to get a basic understanding of the system, main use cases and actors involved in the system. In this phase we came up with 5 use cases and 4 actors that are required for the system. Different use cases and actors are discussed briefly in the following section:

2.1) Use cases

Schedule test: Before a student registers for an exam he/she has to decide about the time and the date he/she wants to take the test. The schedule use case was defined to address this issue. During this use case the system recommends available times and dates for him/her according to his/her preferences.

Register: A student has to register for a test in order to be able to take a test. This use case was defined to address this issue. Authorization information is provided to the student when he/she registers for a test. This way, later on the system can make sure that the legitimate student is taking the test.

Authenticate: The system has to somehow make sure that the rightful student is taking the test. The Authenticate use case was defined to address this issue.

Test Taking: After authentication the user has to be able to take the test. This use case was defined to address this issue.

View Result: After the student finishes his/her test he/she has to be able to view the results of the test. This use case was defined to enable the user to view the results of test right after the test is finished.

2.2) Actors

Student: The actual user of the system is the student agent.

Registrar Clerk: This agent is defined to handle the scheduling and registering of the test.

Test Clerk: This agent is defined to carry out the authorization and test taking processes.

Reporter: This agent is identified to handle reporting of the results to the student.

A detailed description of the use cases can be found in Table 1.

Use case name	Use case description
Schedule test	The main purpose for this use case is to help the user for scheduling his test. The user provides the Registrar clerk agent with his preferred time, date and location for the test. Registrar clerk searches the database for the possible times and provides the user with the most suitable times according to his preference. If there is no test time and location matching the user's preference a message will be given to the user and the user is asked to reschedule his test.
Register	This use case is dedicated for the process of registration. After finding the suitable time and location for the test the user needs to pay for the test. After payment, a username, password and a test registration number will be provided for the user. These pieces of information are needed for authentication when the user wants to take the test.
Authenticate	Before entering the test, an authentication is needed to make sure the user is the correct user for the current time and place. If the authentication is successful the user can start the test otherwise user will be asked for re-entering his authentication information.
Test taking	Test taking use case is dedicated to the processes needed in order to take a test. User starts with the first question which has a medium level of difficulty. After answering based on the performance of user in each step another question will be assigned to the user dynamically until the test is finished. If the user's answer is correct the difficulty level will be increased otherwise it will be decreased or remain the same. A pattern checking process is also performed through this use case to prevent the user from answering questions correctly based on a pattern selected by chance. If a special pattern for selecting the choice is detected the correct choices for next questions are selected in a way which is not mapped to the detected pattern.
View results	After the test is finished user expects to see the results. The reporter agent uses the user's session to check the answers with the correct answers and also makes the best judgment for the user's mark based on the difficulty level of questions. At last, it provides the user with the test report and also the overall score for the test.

Table 1: Description of the use cases of the system in the conceptualization phase

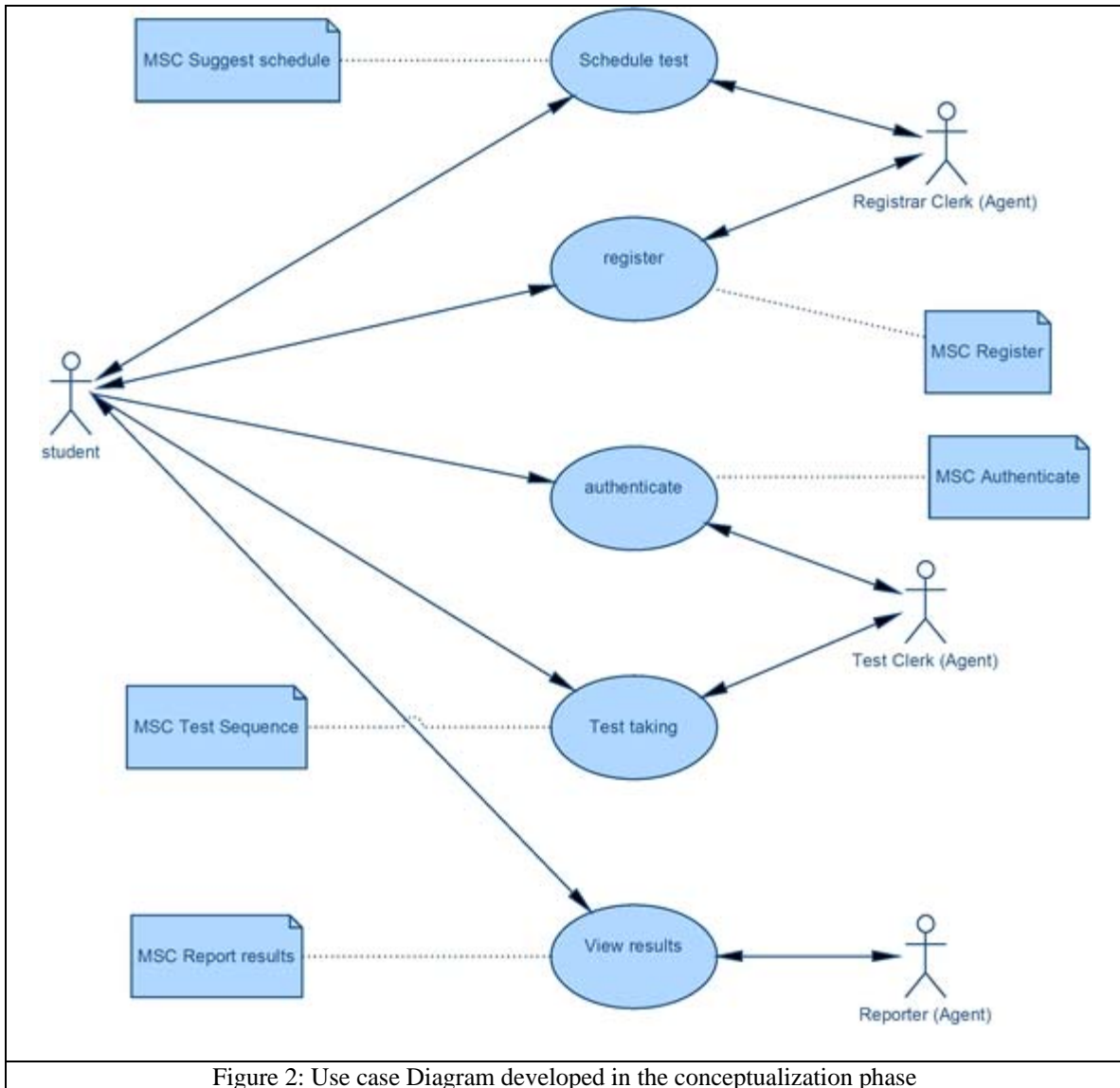
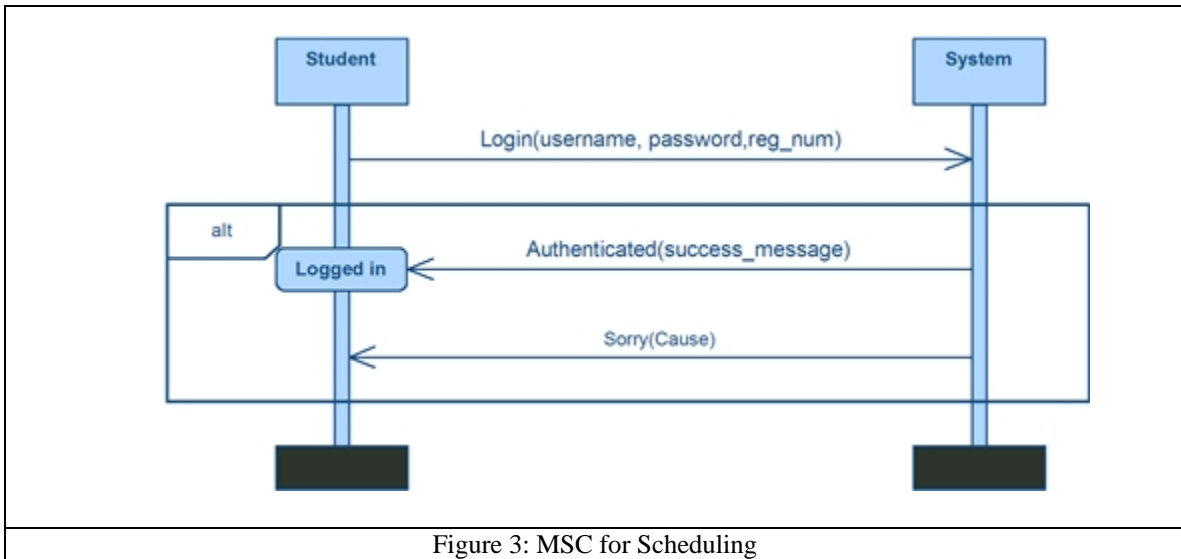


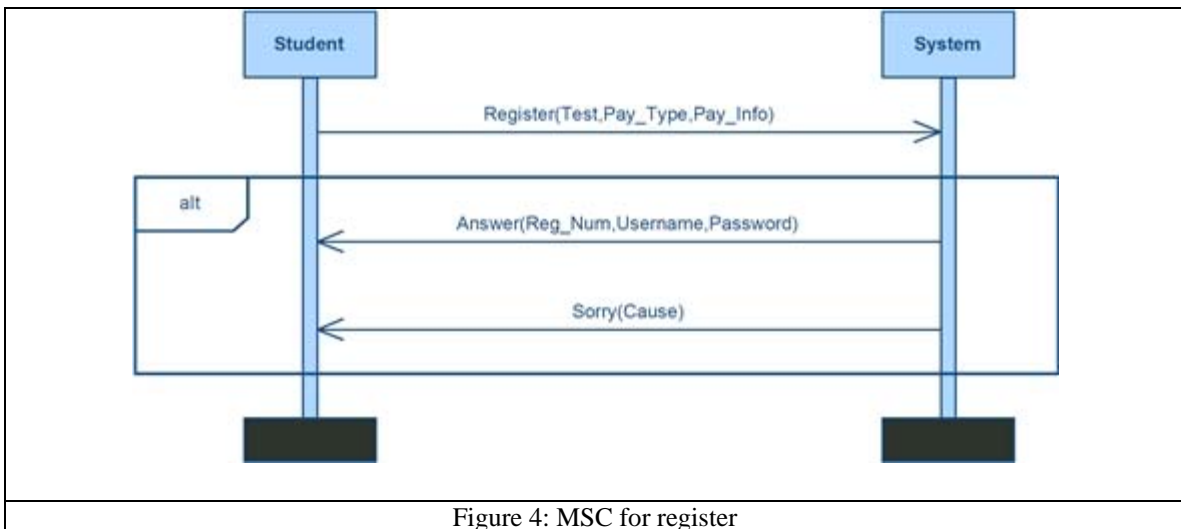
Figure 2: Use case Diagram developed in the conceptualization phase

After defining the use cases and the actors we specified a Message Sequence Chart (MSC) for every use case to better understand the interactions within the system. The MSC for each use case described below:

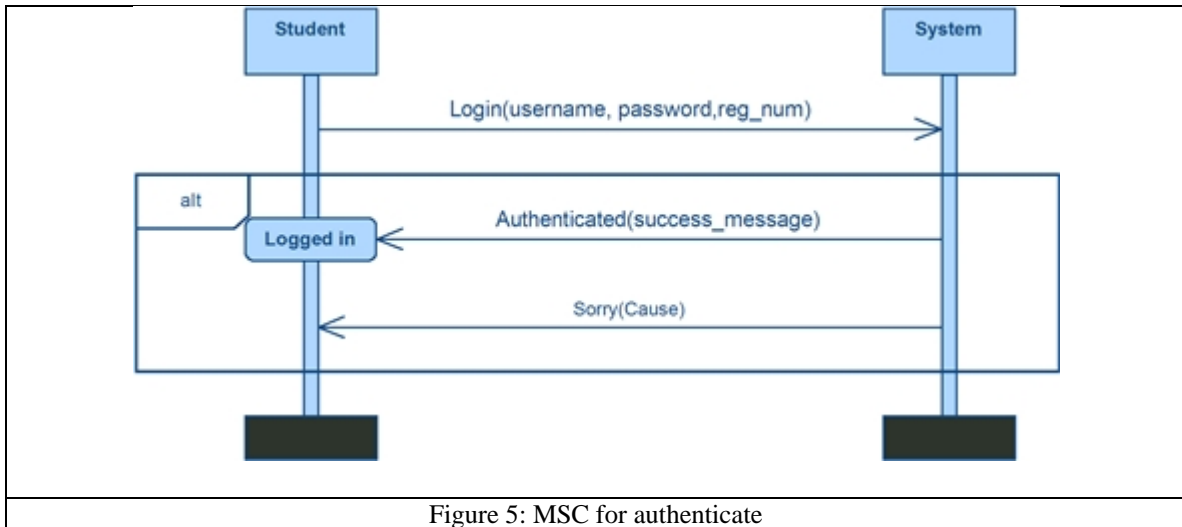
Schedule test: The MSC for this use case is shown in Figure 3. The student specifies his/her preferred time frame with sd (Start Date), ed (End Date), st (Start Time), et (End Time). lc (Location) specifies the preferred location of the test and the “test_area” specifies the field in which the student wants to take the test.



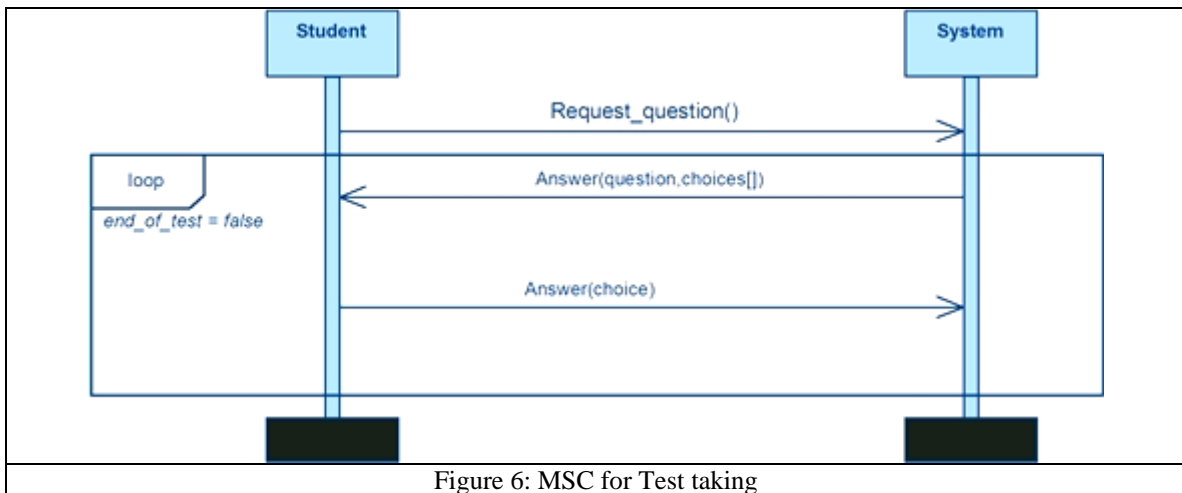
Register: The MSC for this use case is shown in figure 4. The student specifies the test that he/she wants to take with “Test”. The payment type will be specified in “Pay_Type” which can be any type of credit card. Username and Password are the student's authentication information. The “Reg_Num” represents the registration number that the user must keep until the time of the test. The authentication information is only valid for the provided registration number.



Authenticate: In this use case the student provides the registration number (reg_num) that he/she already has along with the authentication information (username and password) provided to him/her previously. The MSC for this use case is illustrated in the Figure 5.



Test taking: In this use case the student is asked one question at a time until the test is finished. “choices[]” represent different alternatives that the student has for answering the question. “choice” represents the user's answer to the question. The MSC for this use case is shown in Figure 6.



View Results: In this use case the student requests for the results of his/her test and a report is shown to him/her. The MSC for this use case is depicted in Figure 7.

In all the use case if the system fails to perform the requested task, apologizes the student and shows him/her the cause of the problem.

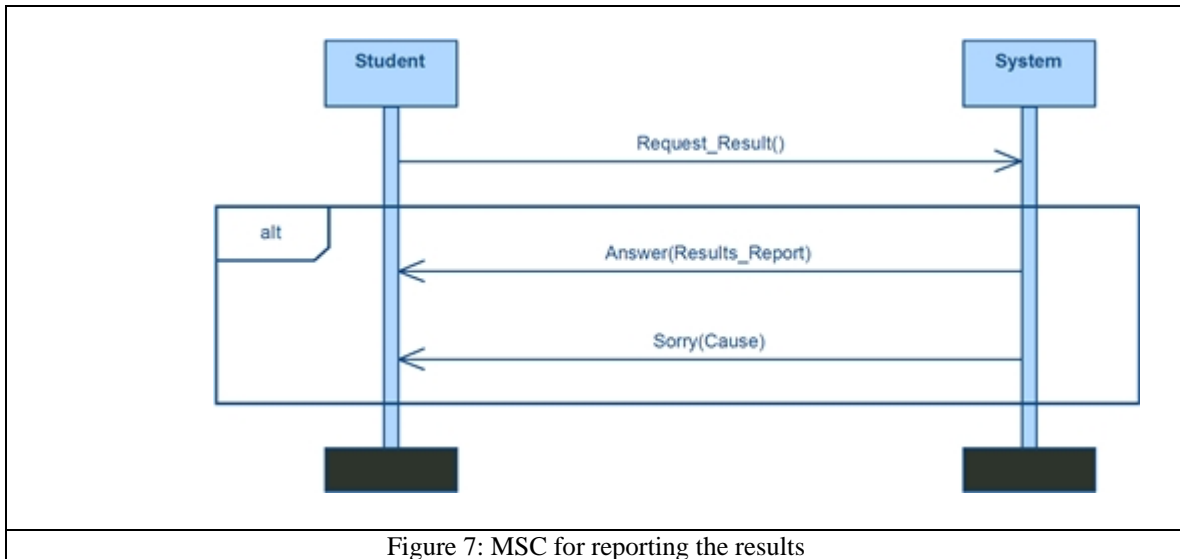


Figure 7: MSC for reporting the results

3) Agent Modeling

To do the agent modeling we took as input the model developed during the previous phase and tried to analyze it further to come up with a complete list of the agents needed for our system. To accomplish this we used the “Internal use case” technique and the conceptual distances existing in the system.

We understood that for the Registrar agent there exists a geographical distance for the information of the student’s account information. So we split the “Register” use case to two use cases, “Registration” and “Process Payment” as shown in Figure 8.

Registration: In this use case the “Registrar” reflects the student's registration information in the system.

Process Payment: In this use case the “Registrar Clerk” processes the payment for the test with the help of the “Bank Clerk”.

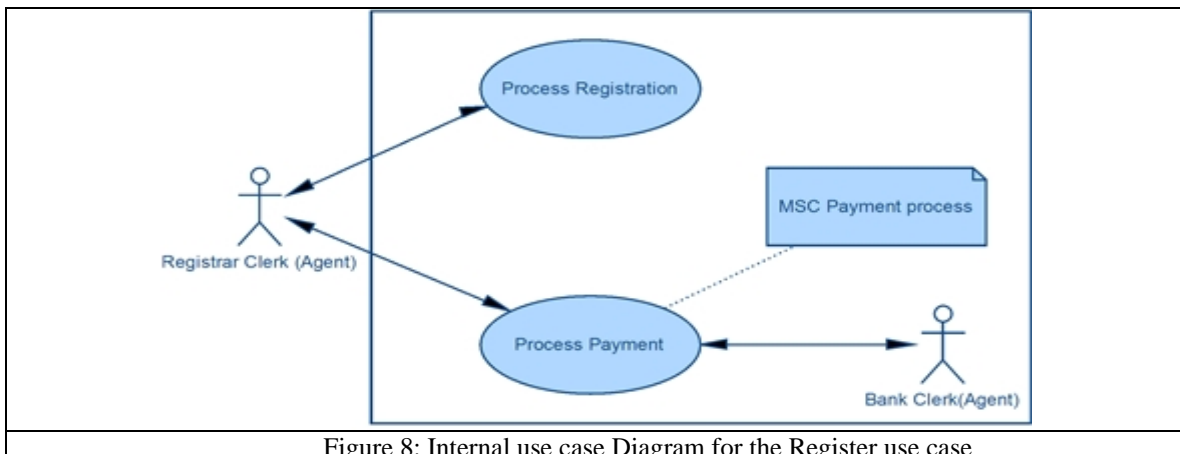


Figure 8: Internal use case Diagram for the Register use case

Table 2 shows the description for these use cases. The MSC for the payment process is shown in Figure 9. “Transaction_Num” is the confirmation number for the transaction that the “Bank Clerk” sends back to the “Registrar Clerk”.

Use case name	Use case description
Process registration	Through this use case, we process all of the steps needed for registration mentioned before in the description of register use case except for payment and finalizing the registration. The main actor for this use case is Registrar clerk.
Process Payment	Here, we process the payment through the payment web service. If the user has enough credit the payment is accomplished and a transaction id is stored in database for the future reference. We also generate the username and password for the user and assign them to the pre specified test to let the user perform the authentication process on correct test date and in correct test place and for the correct test type. The payment process is done by Bank clerk agent.

Table 2: Use case description for “Process registration” and “Process Payment”

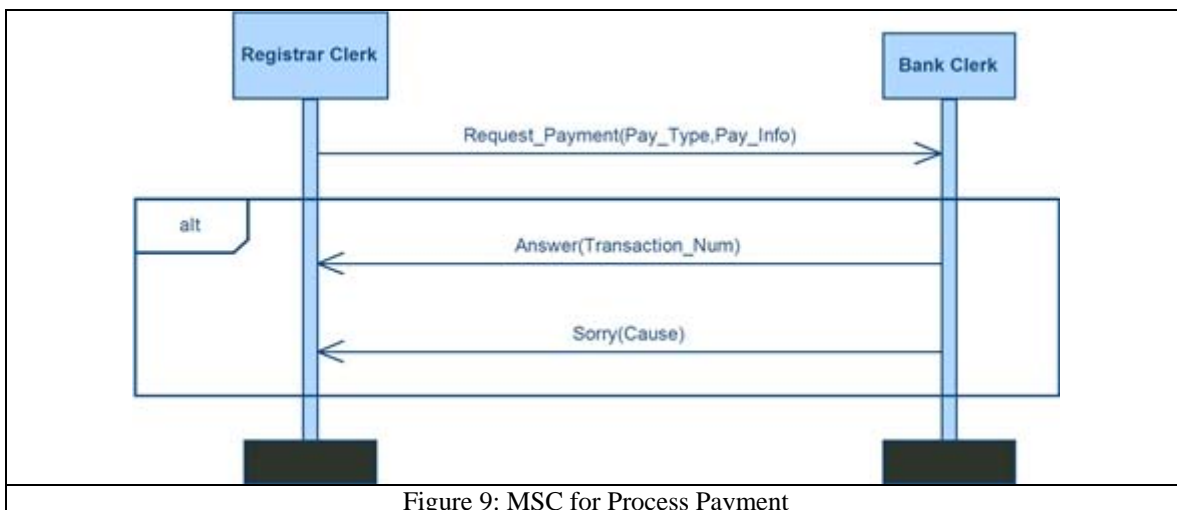


Figure 9: MSC for Process Payment

For the “Test Clerk” we understood that there is knowledge distance for suggestion of the next question. So we split the “Test taking” use case to 2 use cases: “Check answer and Save record” and “Request Next Question” as shown in Figure 10.

Check answer and Save record: In this use case the “Test Clerk” checks the correctness of the answer of the student and maintains the record.

Request Next Question: In this use case the next question is determined by the “Question Provider”. “Question Provider” also tries to minimize the chance for random answers to be correct. Here we also understood that the difficulties of the questions need to be updated over time according to the answers given by different students. So we introduced a new use case for updating the difficulty of the questions. The agent responsible for this is the “Difficulty changer”.

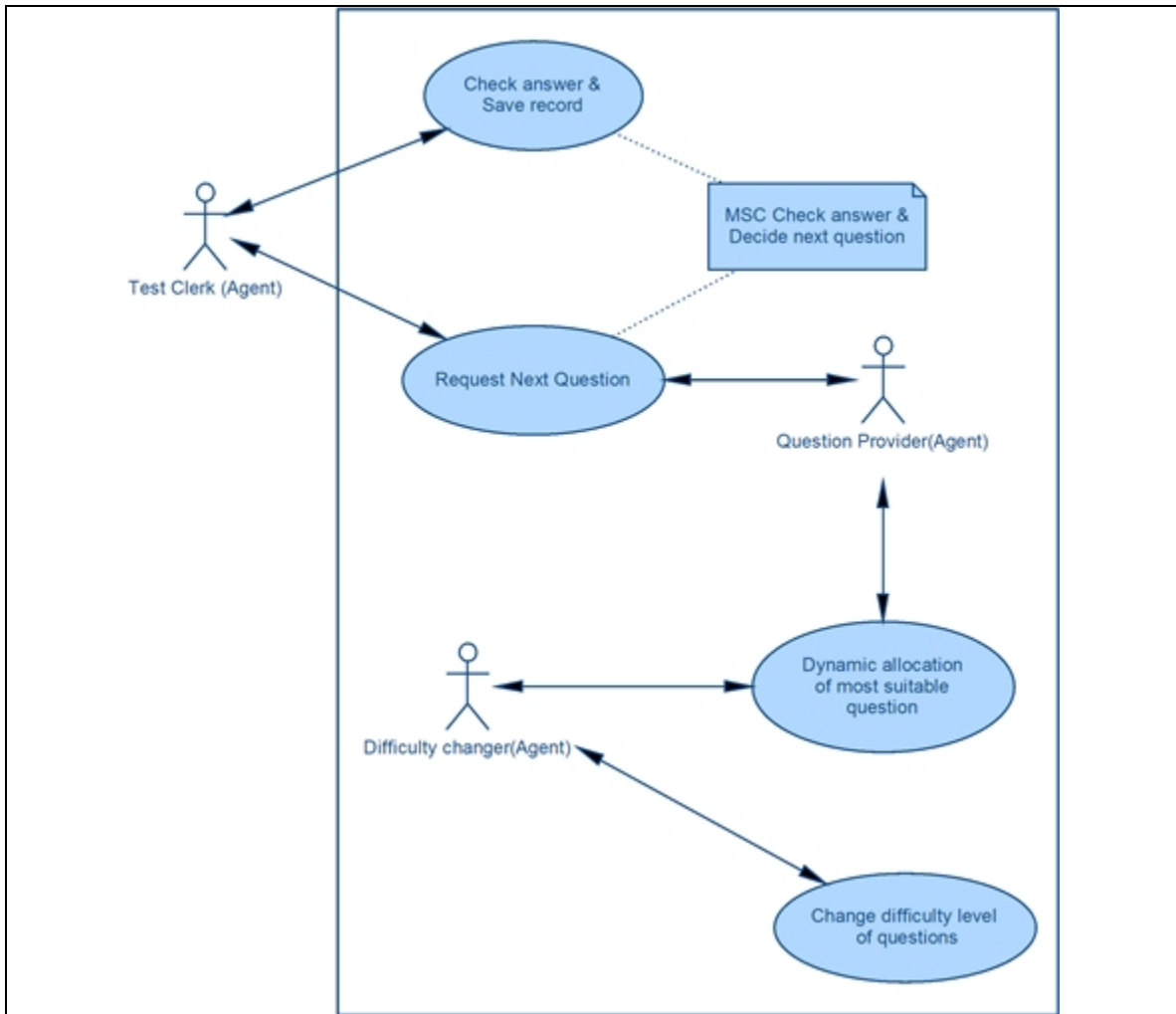


Figure 10: Internal Use case diagram for “Test taking” use case

Table 3 shows the description for these internal use cases. The MSC for this use case is shown in Figure 11. “answered_questions[]” is the answers that the student has given this far.

Use case name	Use case description
Check answer & Save Record	When the user answers the current question. This answer will be passed to Test clerk agent. Test clerk checks the correctness of user’s answer through this use case and saves the record for the evaluation of this question in user’s session.
Request Next question	As said before, the questions are shown to the user in a dynamic way and our test is an adaptive test. Selecting the next question is not the responsibility of Test clerk. Due to the knowledge distance we have here for selection of next question another internal agent is defined named Question provider. Test clerk reads the user’s session each time and passes the user record for answers to the Question provider. After the decision is made by

	Question provider about the next question which best fits the user's performance, the question will be passed to Test clerk through this use case.
Dynamic allocation of most suitable question	Question provider decides about the next question for the user. Question provider needs the reasoning capabilities to allocate a suitable question with a proper difficulty level to the user. After the decision is made about the next question, Question provider asks Difficulty changer agent if the difficulty level for this question is under change or not. If the difficulty level is stable for this question, it is possible to allocate the question to this user. If not, Question provider must select another question. When the suitable question is found, Question provider also retrieves the choices for this question form the database. A pattern checking process should be done here to select the position of correct answer as it is shown to the user. Question provider reviews the previously answers for the user to check if there is any special pattern in giving the correct answer by the user. If a pattern is detected in correct answers, it selects the position for the correct answer in a way which does not match the detected pattern.
Change difficulty level of questions	The difficulty level for questions is assigned to them initially. The initial level might not be suitable for the question. The number of correct or incorrect answers regarding the skill of students who have given those answers is a good measure for understanding the difficulty level of questions. Through this use case, Difficulty changer agent mines the database to measure the difficulty level of question. This use case is initiated each time a new test is completed by the user. The Difficulty changer agent needs reasoning capabilities for estimating the difficulty level of question.

Table 3: Use case description for internal use cases of the Register use case

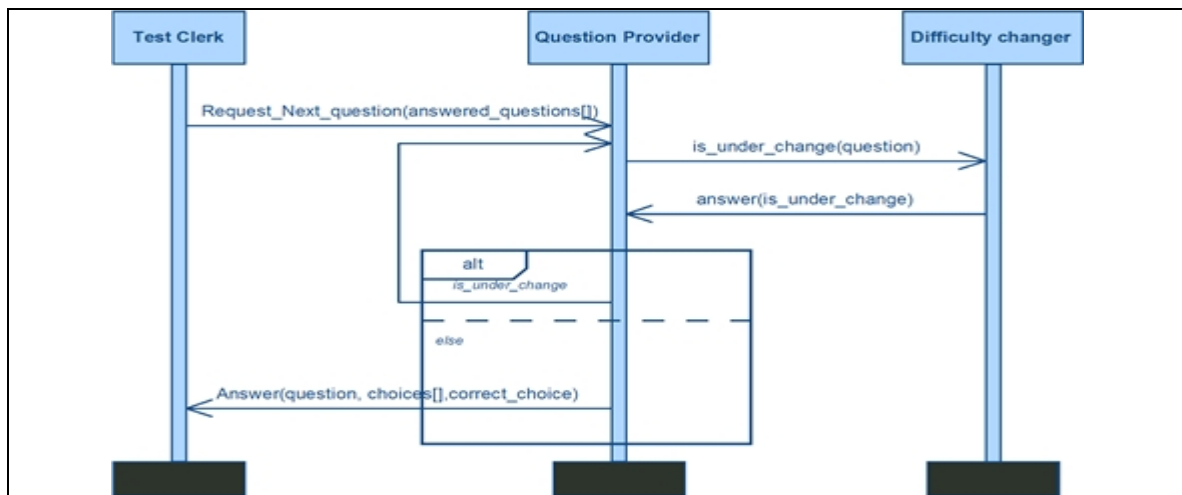


Figure 11: MSC for the "Request Next Question" use case

At this stage we came up with a complete list of all the participant agents in our system. These descriptions were refined through multiple iterations as other models of the systems became more complete. Tables 4-9 show the agents participating in our system and their respective descriptions.

Agent Name	Question Provider
Agent Role	Provider of the next question
Objectives	<ul style="list-style-type: none"> • Allocation of the most suitable question to be asked next from the user based on the responses he/she has given this far. • Prevent the possibility of correct answers by chance.
Description	<ul style="list-style-type: none"> • This agent is included in the model so that the exam can be designed dynamically in run-time and in an adaptive way. • To accomplish this, the user record for the previous answers is obtained through the Test clerk agent from the user's active session. • Then according to the data retrieved in the previous step, it comes up with an initial assumption of the properties of the most proper question to be asked next. • Then the most suitable available question is selected from the Questions Database • This agent needs reasoning capabilities for assigning the best suitable question for the user • Now the agent tries to check the pattern with which the user has answered questions and if it finds out that the user is giving his/her answers correctly by chance it is responsible to provide the Test clerk agent with the next correct choice for selected question in way which does not match the detected pattern. • At this point the agent returns the most suitable question to the "Test Clerk" agent.

Table 4: Agent description for "Question Provider"

Agent Name	Reporter
Agent Role	Result Reporter
Objectives	<ul style="list-style-type: none"> • Giving the most proper mark to the user according to the difficulty of the questions • Reporting the results of the test to user at the end of the test
Description	<ul style="list-style-type: none"> • After the test is finished by the user the agent provides the user with a user interface and asks him/her to decide whether he/she wants to see the results of his/her test • If the user decides that he/she wants to see the results of his/her test the agent accesses the database of the responses that the user has given • At this point the agent assigns a mark to the user according to the difficulty of the asked questions • After calculating the mark the agent provides the user with an interface showing him/her the results of the test and the given mark

Table 5: Agent description for Reporter

Agent Name	Registrar Clerk
Agent Role(s)	User interface for registration, scheduling the test for user, registration of test
Agent Objectives	provision of a friendly user interface, providing the possible test time and locations for the user
Description	<ul style="list-style-type: none"> • This agent has the role of user communication. This agent receives the data needed for scheduling a test from the user. After receiving the data, searches for all possible test date and places which match the user preferences. Returns the list of possible test date and locations to the user. • This agent is also responsible for registration of the test. When the system user selects from the possible options passes a message to this agent and states the preferred test and provides payment information .Registrar clerk tries to register the test for the user. • Registrar Clerk is also in contact with Bank Clerk agent. When this agent receives payment information from the user, asks the Bank Clerk for a transaction. Bank Clerk tries to make a transaction and provides the results. If the transaction is successful, Registrar Clerk registers the user in the requested test and provides the user with a username and password and also a test number which are valid for the duration of the test in the selected location. Besides the registration, this agent also saves the transaction record in test database for future reference. • If the transaction is not successful or for any other reason the registration is not possible it will return the user an error message and will explain the reason for that.

Table 6: Agent description for “Registrar Clerk”

Agent Name	Bank Clerk
Agent Role(s)	Performing the transaction for payment method
Agent Objectives	Ensuring the security of transaction
Description	<ul style="list-style-type: none"> • This agent is responsible for processing the user payment for the test. Bank Clerk is in contact with Registrar Clerk and receives the user payment information from Registrar Clerk. • After receiving the payment information uses the payment web-service to perform the transaction. If the transaction is successful it will provide the Registrars Clerk with the transaction number and if the transaction is not successful it will send a message for Registrars office and will explain the reason for the problem.

Table 7: Agent description for “Bank Clerk”

Agent Name	Test Clerk
Agent Role(s)	User Interface for exams, checking the correctness of user answer, saving the user test record in database, user authentication
Agent Objectives	Providing a friendly user interface, enforcing the test security
Description	<ul style="list-style-type: none"> • Test Clerk agent is the user interface for exams. When the user is taking the test. Test Clerk is also in contact with the Question Provider agent. Each time the user answers a question and moves to the next question, this agent checks the correctness of answer and stores the record for answered question in the session dedicated for the user. • This agent is also responsible for user authentication. When the user tries to login the system, provides this agent with a username, password and a test number. Test Clerk checks the validity of username and password for the given test and adds the username to the session which means changing the status of this user to an active user. • When Test Clerk asks Question Provider for a new question, question provider selects an appropriate question based on the current performance of the user and provides Test Clerk with a new question; question provider also provides the correct choice for the question each time to minimize the possibility of correct answers which are given by chance. • After receiving the new question from Question provides shows the question to the user and repeats this process until the maximum number of questions is achieved.

Table 8: Agent description for “Test Clerk”

Agent Name	Difficulty changer
Agent Role(s)	Changing the difficulty of questions based on the answers given by the students
Agent Objectives	To correct the difficulty level measure for questions
Description	<ul style="list-style-type: none"> • This agent is responsible for updating the difficulty level for each question. The difficulty level of question is specified initially in our database. There is no guarantee that this level of difficulty is suitable for our question. This agent is introduced to improve this measure. • During the time, we obtain more and more records about the answers which are given to specific question in our database. We can use the stored data in our database as a guide to understand the difficulty level of questions. • Difficulty changer iteratively checks the database and by a comparison between the number of correct and incorrect answers to the question decides whether the difficulty level is suitable or not. • The only event which activates this agent is the

	accomplishment of each test. After the end of test, this agent checks and updates the difficulty level of each question involved in the test. And needs some reasoning capabilities to estimate the difficulty of questions based on the initial difficulty and the new correct or incorrect answers to the question.
--	---

Table 9: Agent description for “Difficulty changer”

4) Task Model

At this stage we decomposed the tasks following a top-down approach. As this methodology adopts a component-based approach, we develop the models concurrently and in an iterative style. This way any improvement in one model can help to the improvement of other models in the following iterations. As we completed the task model it helped us more in completing the descriptions of the agents and their objectives.

As a notation we used an and/or tree model. In an and/or tree all possible ways that a task can be accomplished are analyzed. In this notation completion of every task consists of either the completion of all its children tasks or the completion of one of its children tasks. The final task model for our system is illustrated in figure 12. In this model all the tasks are shown in circles, this indicates that in order to complete any task, all its children tasks should be carried out.

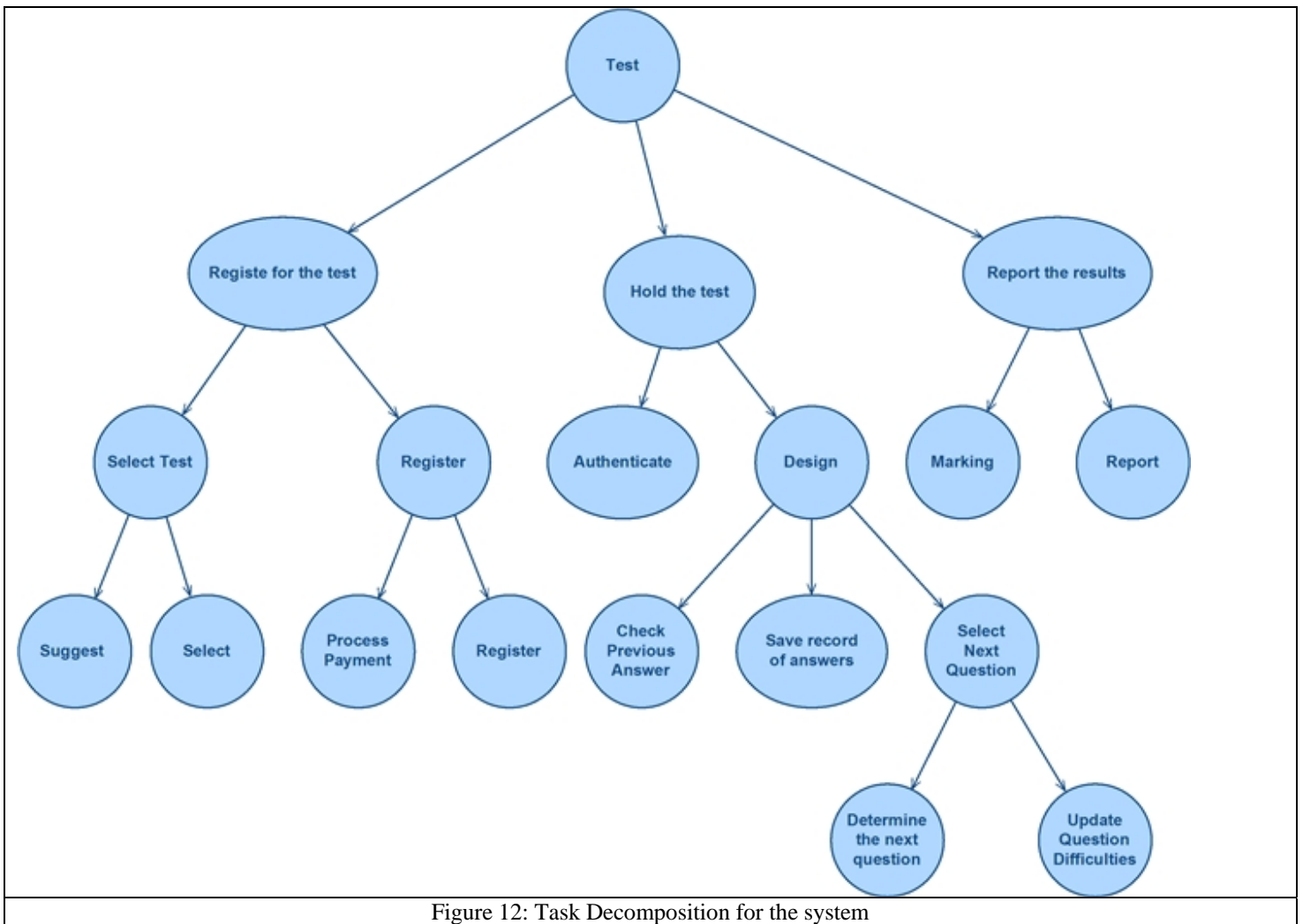


Figure 12: Task Decomposition for the system

5) System Architecture

Analyzing all the developed models, system architecture can be specified. This architecture is represented in Figure 13.

The student takes every possible action through the user interface agents. To implement security, as can be seen in the diagram, “Test Clerk” and “Reporter” agents do not have access to the “Test Database” and they can only access the student's session information. The student's session contains all the temporary information that the agents need to carry out their tasks like the student's login information, answers that the student has given from the beginning of the test, etc. The “Test Database” contains all the persistent information needed for the agents to operate like the questions, test time tables, etc. The “Bank Clerk” processes the payment transaction through the “Bank Web Services”.



Figure 13: System Architecture

6) Data Specification

Data specification diagrams and explanations are provided in this section. Below is the general view of tables and relationships. Some tables are repeated in this view just to make the diagram clearer.

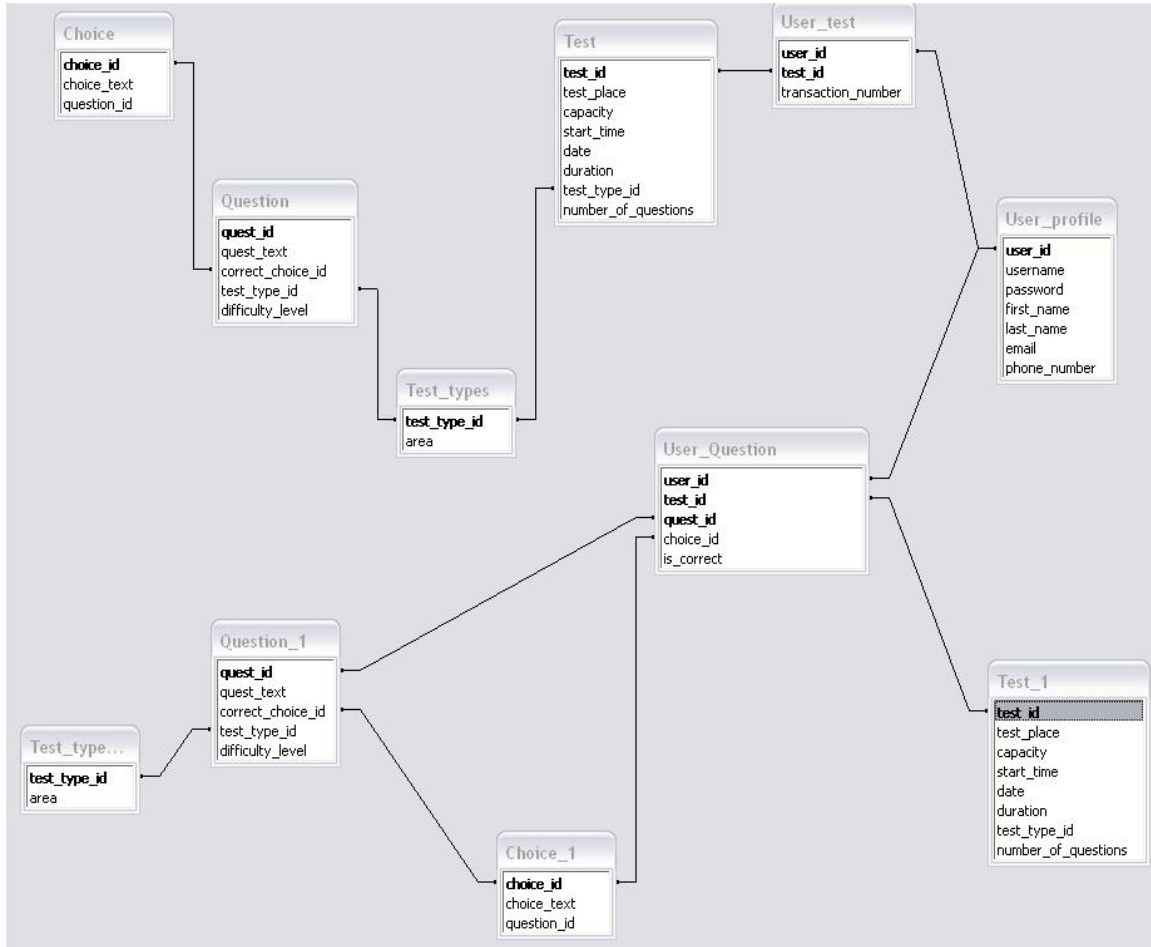


Figure14. Table relationships for test database

Basically, there are 7 tables in our test database which are: Test, Test_types, Question, Choice, User_profile, User_test and User_question. Now we explain each table and talk about the data types.

Test table:

This table contains data which is needed to define a test. Each row of this table defines a test section. Test_id is the primary key for this table. test_place is the location for the test. Capacity is the maximum number of students who can register for the test. Another field which needs more explanation is test_type_id. This field is a foreign key from the test_types table which relates the test to the corresponding test type.

	Field Name	Data Type
🔑	test_id	Number
	test_place	Text
	capacity	Number
	start_time	Date/Time
	date	Date/Time
	duration	Number
	test_type_id	Number
	number_of_questions	Number

Figure15. Fields and field types for test table

Test_types table:

This table contains the data for existing test types in our database. test_type_id is the primary key for this table and area specifies the course area for this test type.

	Field Name	Data Type
🔑	test_type_id	Number
	area	Text

Figure16. Fields and field types for test_types table

Question table:

This table contains data about questions. question_id is the primary key for this table. question_text is the question itself. Each question is related to the choice table through the correct_choice_id foreign key which specifies the correct answer for this question. Each question also belongs to a test type and it has a difficulty level.

	Field Name	Data Type
🔑	quest_id	Number
	quest_text	Text
	correct_choice_id	Number
	test_type_id	Number
	difficulty_level	Number

Figure17. Fields and field types for Question table

Choice table:

This table stores the choices for multiple choice questions. choice_id is the primary key for this table. choice_text is the textual representation of the choice and the question_id specifies the related question for this choice.

	Field Name	Data Type
🔑	choice_id	Number
	choice_text	Text
	question_id	Number

Figure18. Fields and field types for Choice table

User_profile table:

This table plays the role of user's profile. User_id is the primary key for this table. username and password are the same username and password which are given to the user to take the test. And the rest of fields are clear.

	Field Name	Data Type
🔑	user_id	Number
	username	Text
	password	Text
	first_name	Text
	last_name	Text
	email	Text
	phone_number	Text

Figure19. Fields and field types for User_profile table

User_test table:

This table holds the relationship between user_profile and test tables. The primary key for this table is the combination of user_id and test_id. Both of these fields are foreign keys from external tables. transaction_number for user payment is also stored in this table.

	Field Name	Data Type
🔑	user_id	Number
🔑	test_id	Number
	transaction_number	Text

Figure20. Fields and field types for Choice table

User_question table:

This table holds all of the given answers from users to the questions. And also keeps the correctness of that answer. The data in this table can be used in future to update the difficulty level of questions.

	Field Name	Data Type
🔑	user_id	Number
🔑	test_id	Number
🔑	quest_id	Number
	choice_id	Number
	is_correct	Yes/No

Figure21. Fields and field types for Choice table

The primary key is the combination user_id, test_id and quest_id. All of these fields are foreign keys from external tables. choice_id is the selected choice by the user for the question. And is_correct Boolean field determines the correctness of that answer.

User's session:

Besides the data which is stored in the database there is another means of data sharing in our system. User interface agents have access to user's session. While a student is taking the test, his record of given answers is not saved in the data base. It is just stored in user's session. After the test is finished the session for the student will be destroyed.

7) Communication specification

In this section of document we focus on the messages interchanged between agents and the content of each message. The event flow diagram in figure 22 summarizes the message passing of agents. The SOAP protocol is used by Bank Clerk agent to interact with the web service. For inter agent interactions the format of message is XML. Each interaction consists of a "request" message and a "return message". The SOAP protocol is also used for interactions between agents.

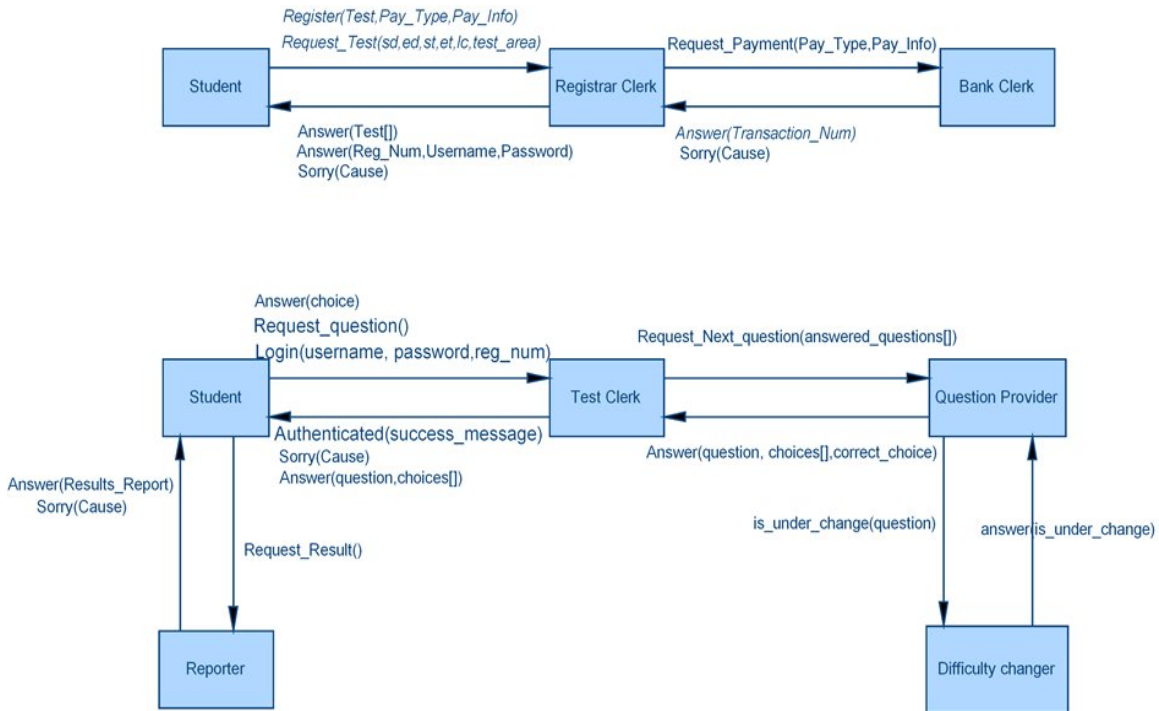


Figure22. Messages passed between agents

Student-Registrar Clerk interaction for scheduling:

The format of request message from student to Registrar Clerk agent is as below:
 Request_test(sd, ed, st, et, lc, test_area)

The parameters for Request_test message are:

- sd = start date
- ed = end date
- st = start time
- et = end time
- lc = location
- test_area = the course area for the test

The data type for these input parameters are listed below:

parameter	Data type
sd	date
ed	date
st	time
et	time
lc	string
test_area	string

The response for this message is either an array of Test entities or a “sorry” message stating the result for which the search had not been successful. Test entity is described in XML notation as below:

```

<Test>
  <test_location> String </ test_location >
  <test_date> date </ test_date >
  <start_time>time</ start_time >
  <duration>long</duration>
  <number_of_questions>integer</ number_of_questions>
</Test>

```

Return message for this interaction is “answer(Test[])” or “Sorry(cause)”
 The return message Answer contains an array of Test entities described above. If the search for finding suitable tests is not successful the result will be the “sorry” message. The return parameter for sorry message is a String.

Student-Registrar Clerk interaction for registration:

The request message for this interaction is Request_test(Test, Pay_type, Pay_info). The Test parameter here is the same as the Test parameter in the previous message. pay type is String and Pay_info is as follows:

```

<Pay_info>
  <credit_type>string</credit_type>
  <credit_number>long</credit_number>
  <pin_code>integer</pin_code>
</Pay_info>

```

The return message for this interaction is “Answer(reg_num, username, password)” contains the username, password and the registration number. Username and password are strings and registration number is an integer. There is an alternative message for the case that registration is not successful. The message name is Sorry and the data format for this message is the same as Sorry message in previous interaction. The format of this return message is the same wherever it is used. So, we do won’t mention this message for next interactions.

Registrar Clerk -Bank Clerk interaction for payment:

The request message is “Request_payment(Pay_type, Pay_info)” the format of Pay_type and Pay_info fields is the same as what we described before.

The return message is “Answer(transaction_num)” and the parameter type of transaction_num is integer.

Student -Test Clerk interaction for authentication:

the request message is “login(username, password, reg_num)” and the return message is “Authenticated(success_message)”. Username and password are Strings. reg_num is the test registration number and the type of this field is integer.

Student -Test Clerk interaction for taking the test:

the request message which is sent by student to test clerk is “Request_question()” this message works as an event to request for the next question and does not have any parameter. The return message is answer question, choices. question is text and choices is an array of text. This message itself is the request message from Test Clerk to the student. The return message for this message is “Answer(choice)”. This message is sent from student to the Test Clerk. The input parameter for this message is also string.

Test Clerk –Question Provider interaction:

The request message from Test Clerk to Question provider is:

Request_Next_Question(answered_questions[])

The input value for this message is an array of answered_question. The structure of answered_question is described in XML format here.

```
<answered_question>
    <question_text>string</question_text>
    <choice1>string</choice1>
    <choice2>string</choice2>
    <choice3>string</choice3>
    <choice4>string</choice4>
    <correct_choice>integer</correct_choice>
    <selected_choice>integer</selected_choice>
</answered_question>
```

The return message for this interaction is the message Answer(question, choices[], correct_choice). Question is a string, choices is an array of string and correct_choice is an integer.

Difficulty manager –Question Provider interaction:

the request message from query provider is is_under_change(question). The only parameter for this message is question with the field type of string. The return message is answer(is_under_change) is_under_change which is the return value of this message has the data type of boolean.