Software Requirement Specification

Version 1.0

Date :- 30 October 2018

### Online student and assessment management

Naman Gupta

(1611981442)

Submitted in partial fulfillment

Of the requirements of

CSL3306 Software Engineering

**Table of Contents**

**I. Introduction**

A. Purpose

B. Scope

C. Definition, Acronyms, or Abbreviations

D. References

E. Overview

**II. General Description**

A. Product Perspective

B. Product Functions

C. User Characteristics

D. General Constraints

E. Assumptions

**III. Functional Requirements**

A. Description

B. Functional Diagrams

**IV. Non Functional Requirements**

A. Hardware Requirements

B. Software Requirements

C. Other Requirements

**­V. System Architecture**

A. Frameworks used

A.1 Java

A.2 JDBC

A.3 JSP

**VI. Appendices**

A. Activity Diagram

B. State Diagram

# I. Introduction

## A. Purpose

The purpose of this document is to inculcate a detailed description of the Online student and assessment management system. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external information.

***B. Scope***

This project particularly aims towards improving the ways in which students are evaluated and tested. Furthermore, both teachers and parents of a particular student can easily look upon the progress of the student.

Furthermore, this project can subvert and help students to improve and shuffle them on to the next level.

***C. Definition, Acronyms, or Abbreviations***

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Author | Person submitting an article to be reviewed. In case of multiple authors, this term refers to the *principal author*, with whom all communication is made. |
| Database | Collection of all the information monitored by this system. |
| Software Requirements Specification(SRS) | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Reader | Anyone visiting the site to read articles. |
| Field | A cell within a form. |
| HTML (Hyper Text Mark-Up Language) | Markup Language used to create web-pages. |
| MySQL | Database management system used for the project. |
| Java | Object oriented programming language. |
| JDBC | Java Database Connectivity. |
| JSP | Java Servlet Programming. |
| User | Reviewer or Author. |

***D. References***

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

https://ieeexplore.ieee.org/document/8355079

***E. Overview***

In the next chapter, the Genenral Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third and fourth chapters ,functional requirements and non-functional requirements, which are requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Fifth and sixth chapters focus on the System Architecture and System Models which define the structure, behavior, and modular views of a system.

***Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.***

**II.General Description**

***A. Project Perspective***

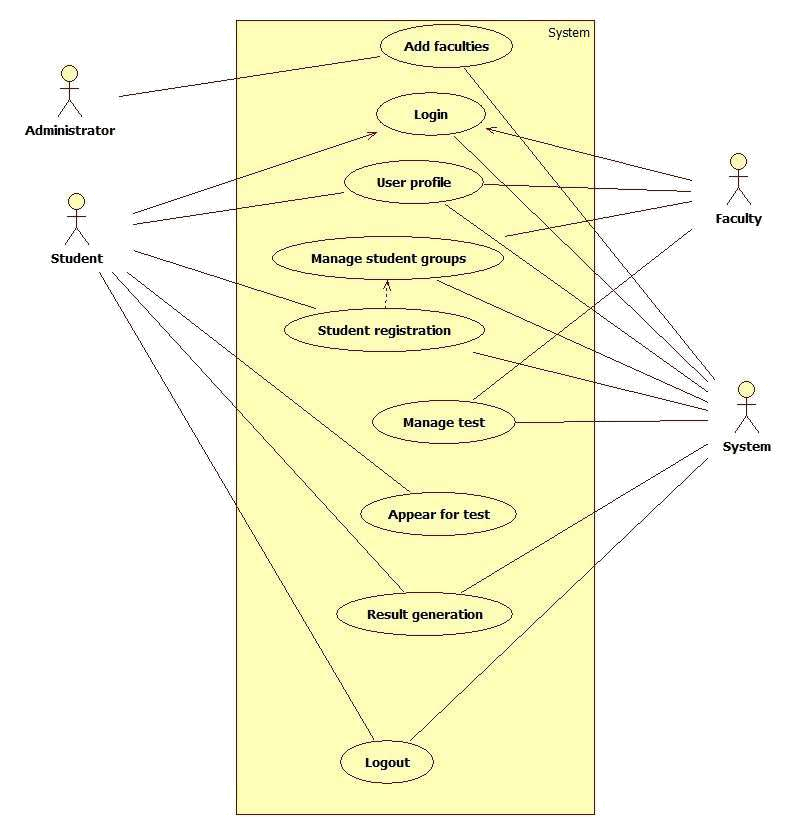
In an era known as the society of technology and knowledge, where lifelong learning is a way of life, it is important that educational institutions have as a priority the goal of finding effective ways of providing new learning opportunities according to their environment, student characteristics, teacher training, economic crisis and advancing technology in an effort to make learning more efficient, recognizing the need and the opportunities, henceforth establishing and developing Online student and assessment management system in order to engage and motivate students according to their necessities. This SRS presents the outcomes and synthesizes the insights of various student entities and complaints. But the main objective of is to present our strategies, vision and goals when we talk about electronic learning independently of is topology.

***B. Product Functions***

The System shall be able to produce following functions :-

* Create student accounts.
* Create admin accounts.
* Create, update Courses.
* Maintain Student Reports.
* Current Preparation Progress.

**C. User Characteristics**

****

**D. General Constraints**

This product will work on client-server architecture. It will be require an internet server and which will be able to run Java application. The product should be support some commonly used browsers such as Internet Explorer, Mozilla Firefox. External interfaces include key board and mouse, enabling navigations across the screens.

**E.Assumptions**

Some points of Assumptions are : -

* All the users (ie students and teachers) have a scintillating internet connection.
* The Desktop’s/ Laptop’s and other hardware reqirements are already present with the user.
* JRE/ JVM are already installed in the computer systems.
* All systems must support atleas dll 365 netframework beta. (Won’t work on dos based systems).

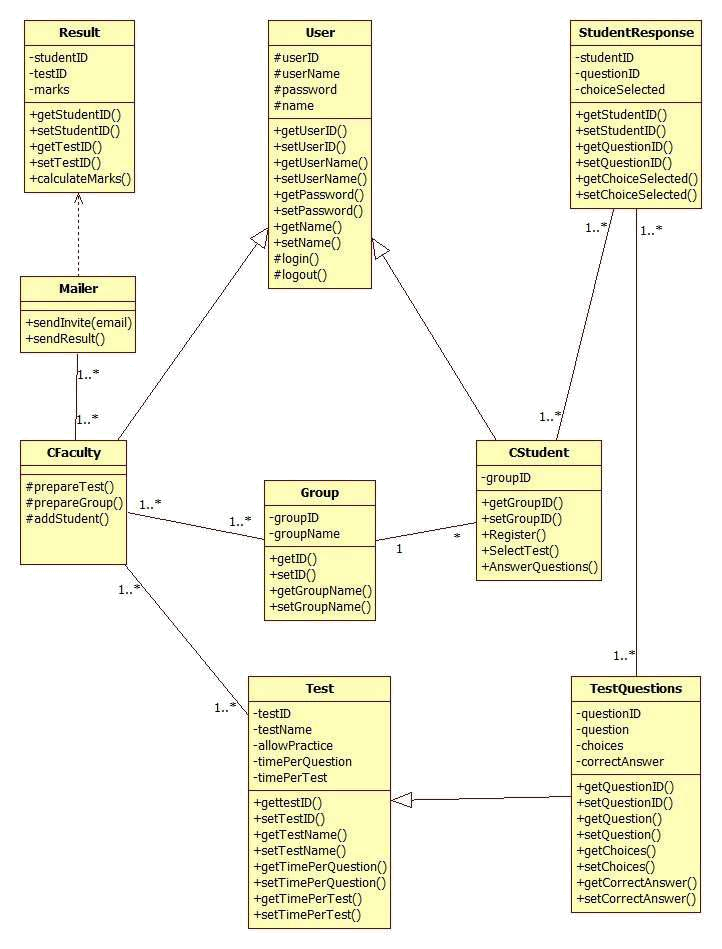
**III Functional Requirements**

**A. *Description***

The Online student accessment and management system is developed for handling the activities for various users such as student, teachers, parents. Every teacher and student, should have laptop with wireless internet connection.

This System should Help option in which how to operate the system should be explained. Also hard copy of this document should be given to the user in item form server.

**B. *Functional Diagram***

****

**IV Non Functional Requirements**

***A.Hardware Requirements***

* The application demands that all the PCs must be present in the internet.
* Pc should be sufficiently fast with adequate memory at least 64 MB RAM and 2 GB hard –disk space is required to run this application.
* Screen resolution of at least 800\*600 required to properly view the screen.
* It should be support the printers.

***B. Software Requirements***

* Any Window Operating System. (No Dos) *.*
* Java must be installed. For the database handling MYSQL and SQLite manager must be installed.
* The final application must be packaged in a set up program, so that the product can be easily installed on the clients-machine.

***C. Other Requirements***

***1. Security***

Application will allow only valid users to access the system. Access to any will application resource depend upon user’s designation. There are two types

of users namely Administrator and Student . Security is based upon the individual user ID and Password.

**2. Reliability**

The application should be highly reliable and it should generate all the updated information in correct order.

**3. Availability**

System will be available around the clock except for the time required for the back up of data.

**4. Maintainability**

The installation and operation manual of student management system will be provided to user.

**5. Portability**

The application should be portable on any windows based system

**V. System Architecture**

**A. *Frameworks used***

*A.1 Java*

Java is a programming language created by James Gosling from Sun Microsystems (Sun) in 1991. The target of Java is to write a program once and then run this program on multiple operating systems. The first publicly available version of Java (Java 1.0) was released in 1995. Sun Microsystems was acquired by the Oracle Corporation in 2010. Oracle has now the steermanship for Java. In 2006 Sun started to make Java available under the GNU General Public License (GPL). Oracle continues this project called OpenJDK.

Over time new enhanced versions of Java have been released. The current version of Java is Java 1.8 which is also known as Java 8.

Java is defined by a specification and consists of a programming language, a compiler, core libraries and a runtime (Java virtual machine) The Java runtime allows software developers to write program code in other languages than the Java programming language which still runs on the Java virtual machine. The Java platform is usually associated with the Java virtual machine and the Java core libraries.

The Java language was designed with the following properties:

* Platform independent: Java programs use the Java virtual machine as abstraction and do not access the operating system directly. This makes Java programs highly portable. A Java program (which is standard-compliant and follows certain rules) can run unmodified on all supported platforms, e.g., Windows or Linux.
* Object-orientated programming language: Except the primitive data types, all elements in Java are objects.
* Strongly-typed programming language: Java is strongly-typed, e.g., the types of the used variables must be pre-defined and conversion to other objects is relatively strict, e.g., must be done in most cases by the programmer.
* Interpreted and compiled language: Java source code is transferred into the bytecode format which does not depend on the target platform. These bytecode instructions will be interpreted by the Java Virtual machine (JVM). The JVM contains a so called Hotspot-Compiler which translates performance critical bytecode instructions into native code instructions.
* Automatic memory management: Java manages the memory allocation and de-allocation for creating new objects. The program does not have direct access to the memory. The so-called garbage collector automatically deletes objects to which no active pointer exists.

The Java syntax is similar to C++. Java is case-sensitive, e.g., variables called myValue and myvalue are treated as different variables.

A.2 JDBC (Java Data Base Connectivity)

The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database.

JDBC helps you to write Java applications that manage these three programming activities:

1. Connect to a data source, like a database
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to your query

JDBC includes four components:

1. **The JDBC API** —  The JDBC API provides programmatic access to relational data from the Java programming language. Using the JDBC API, applications can execute SQL statements, retrieve results, and propagate changes back to an underlying data source. The JDBC API can also interact with multiple data sources in a distributed, heterogeneous environment.

The JDBC API is part of the Java platform, which includes the *Java Standard Edition* (Java SE ) and the *Java Enterprise Edition* (Java EE). The JDBC 4.0 API is divided into two packages: java.sql and javax.sql. Both packages are included in the Java SE and Java EE platforms.

1. **JDBC Driver Manager** —  The JDBC DriverManager class defines objects which can connect Java applications to a JDBC driver. DriverManager has traditionally been the backbone of the JDBC architecture. It is quite small and simple.

The Standard Extension packages javax.naming and javax.sql let you use a DataSource object registered with a *Java Naming and Directory Interfaces* (JNDI) naming service to establish a connection with a data source. You can use either connecting mechanism, but using a DataSource object is recommended whenever possible.

1. **JDBC Test Suite** —  The JDBC driver test suite helps you to determine that JDBC drivers will run your program. These tests are not comprehensive or exhaustive, but they do exercise many of the important features in the JDBC API.
2. **JDBC-ODBC Bridge** —  The Java Software bridge provides JDBC access via ODBC drivers. Note that you need to load ODBC binary code onto each client machine that uses this driver. As a result, the ODBC driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in a three-tier architecture.

This Trail uses the first two of these these four JDBC components to connect to a database and then build a java program that uses SQL commands to communicate with a test Relational Database. The last two components are used in specialized environments to test web applications, or to communicate with ODBC-aware DBMSs.

A.3 *JSP*

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

**JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offers several advantages in comparison with the CGI.**

* Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.
* JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
* JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including **JDBC, JNDI, EJB, JAXP,** etc.
* JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

**VI. Appendices**

**A. Appendix 1- Activity diagram manual**

**UML Activity Diagram**

**Overview:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system.

Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

**Purpose:**

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

So the purposes can be described as:



Draw the activity flow of a system.



Describe the sequence from one activity to another.



Describe the parallel, branched and concurrent flow of the system.

**Basic Activity Diagram Symbols and Notations**

**Action states**

Action states represent the noninterruptible actions of objects. You can draw an action state in SmartDraw using a rectangle with rounded corners.



**Action Flow**

Action flow arrows illustrate the relationships among action states.

**Object Flow**

Object flow refers to the creation and modification of objects by activities. An object flow arrow from

an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.

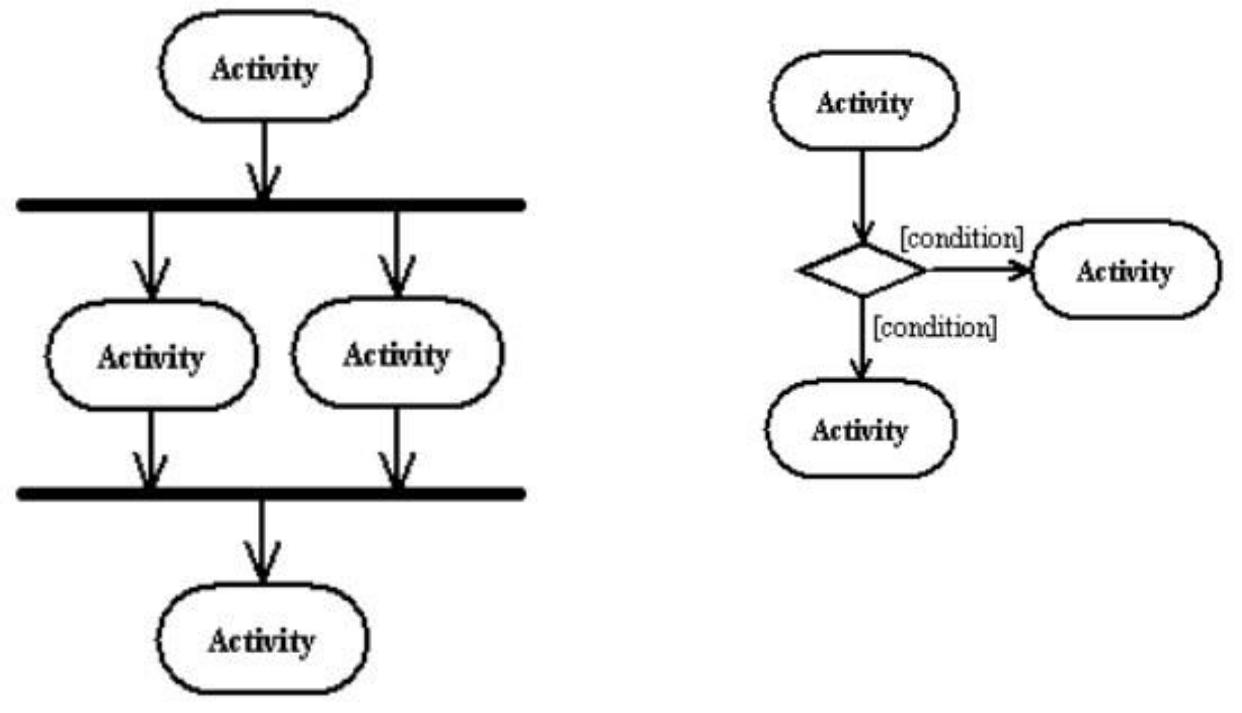
**Initial State** **Final State**

A filled circle followed by an arrow represents the initial action state.

An arrow pointing to a filled circle nested inside another circle represents the final action state.

**Branching**

A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



**Synchronization**

A synchronization bar helps illustrate parallel transitions.

Synchronization is also called forking and joining.

**Swimlanes**

Swimlanes group related activities into one column.

**Where to use Activity Diagrams?**

The basic usage of activity diagram is similar to other four UML diagrams. The specific usage is to model the control flow from one activity to another. This control flow does not include messages. The activity diagram is suitable for modeling the activity flow of the system. An application can have multiple systems. Activity diagram also captures these systems and describes flow from one system to another. This specific usage is not available in other diagrams. These systems can be database, external queues or any other system. Now we will look into the practical applications of the activity diagram. From the above discussion it is clear that an activity diagram is drawn from a very high level. So it gives high level view of a system. This high level view is mainly for business users or any other person who is not a technical person. This diagram is used to model the activities which are nothing but business requirements. So the diagram has more impact on business understanding rather implementation details.

Following are the main usages of activity diagram:



Modelling work flow by using activities.



Modelling business requirements.



High level understanding of the system's functionalities.



Investigate business requirements at a later stage.

**A.2 Appendix 2- StateChart Diagram manual**

**UML Statechart Diagram**

**Overview:**

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A Statechart diagram describes a state machine. Now to clarify it state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. Activity diagram explained in next chapter, is a special kind of a Statechart diagram. As Statechart diagram defines states it is used to model lifetime of an object.

**Purpose:**

Statechart diagram is one of the five UML diagrams used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So Statechart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So the most important purpose of Statechart diagram is to model life time of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. But the main purpose is to model reactive system.

Following are the main purposes of using Statechart diagrams:



To model dynamic aspect of a system.



To model life time of a reactive system.



To describe different states of an object during its life time.



Define a state machine to model states of an object.

**Basic Statechart Diagram Symbols and Notations**

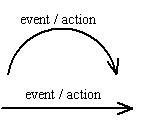
**States**

States represent situations during the life of an object. You can easily illustrate a state in SmartDraw by using a rectangle with rounded corners.



**Transition**

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.



**Initial State**

A filled circle followed by an arrow represents

the object's initial state.



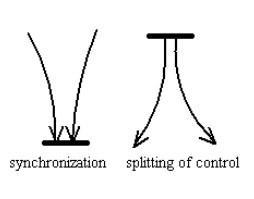
**Final State**

An arrow pointing to a filled circle nested inside another circle represents the object's final state.



**Synchronization and Splitting of Control**

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.



**How to draw Statechart Diagram?**

Statechart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately.

Statechart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs. Before drawing a Statechart diagram we must have clarified the following points:



Identify important objects to be analyzed.



Identify the states.



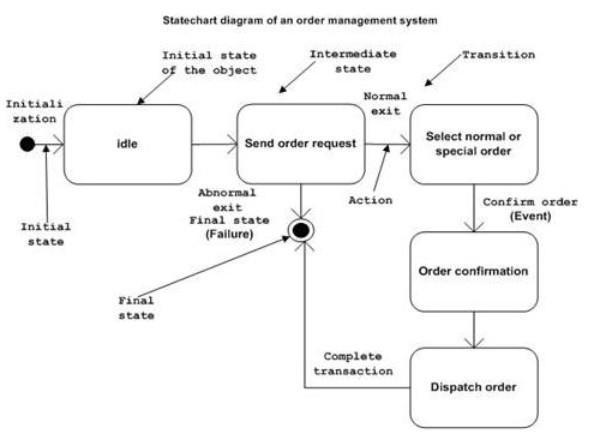
Identify the events.

The following is an example of a Statechart diagram where the state of *Order* object is analyzed.

The first state is an idle state from where the process starts. The next states are arrived for events like *send request*,*confirm request*, and *dispatch order*.

These events are responsible for state changes of order object. During the life cycle of an object (here order object) it goes through the following states and there may be some abnormal exists also. This abnormal exit may occur due to some problem in the system. When the entire life cycle is complete it is considered as the complete transaction as mentioned below.

The initial and final state of an object is also shown below.



**Where to use Statechart Diagrams?**

From the above discussion we can define the practical applications of a Statechart diagram. Statechart diagrams are used to model dynamic aspect of a system like other four diagrams disused in this tutorial. But it has some distinguishing characteristics for modeling dynamic nature.

Statechart diagram defines the states of a component and these state changes are dynamic in nature. So its specific purpose is to define state changes triggered by events. Events are internal or external factors influencing the system.

Statechart diagrams are used to model states and also events operating on the system. When implementing a system it is very important to clarify different states of an object during its life time and statechart diagrams are used for this purpose. When these states and events are identified they are used to model it and these models are used during implementation of the system.

If we look into the practical implementation of Statechart diagram then it is mainly used to analyze the object states influenced by events. This analysis is helpful to understand the system behaviour during its execution. So the main usages can be described as:



To model object states of a system.



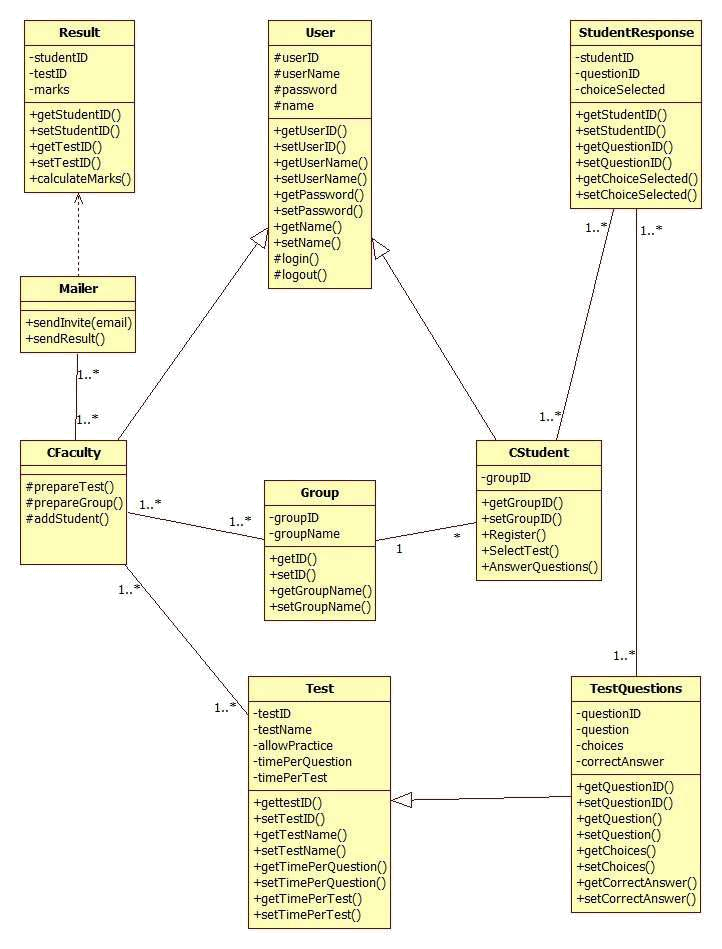
To model reactive system. Reactive system consists of reactive objects.



To identify events responsible for state changes.



Forward and reverse engineering.

****