**OOAD'S MINI PROJECT ON**

**EXAM REGISTRATION SYSTEM**

**Collaboration by,**



**RA1511008010714**

**RA1511008010724**

**RA1511008010727**

**EXAM REGISTRATION SYSTEM**

**AIM:**

To create a system to perform the Exam Registration system

**PROJECT DESCRIPTION:**

Exam Registration system.is used in the effective dispatch of registration form to all of the students. This system adopts a comprehensive approach to minimize the manual work and schedule resources, time in a cogent manner. The core of the system is to get the online registration form (with details such as name, reg.no etc.,) filled by the student whose testament is verified for its genuineness by the Exam Registration System with respect to the already existing information in the database. This forms the first and foremost step in the processing of exam application. After the first round of verification done by the system, the information is in turn forwarded to the Exam Controller. The application is then processed manually based on the report given by the system. The system also provides the student the list of exam dates.The controller will be provided with fees details to display the current status of application to the student, which they can view in their online interface. After all the necessary criteria has been met, the original information is added to the database and the hall ticket is sent to the student.

**REQUIREMENT SPECIFICATION:**

**INTRODUCTION**

Exam Registration System is an interface between the Student and the Exam Controller responsible for the Issue of Hall Ticket. It aims at improving the efficiency in the Issue of Hall ticket and reduces the complexities involved in it to the maximum possible extent.

**PURPOSE**

If the entire process of 'Issue of Hall ticket' is done in a manual manner then it would takes several days for the hall ticket to reach the student. Considering the fact that the number of students for hall ticket is increasing every year, an Automated System becomes essential to meet the demand. So this system uses several programming and database techniques to elucidate the work involved in this process. As this is a matter of National Security, the system has been carefully verified and validated in order to satisfy it.

**SCOPE**

• The System provides an online interface to the user where they can fill in their personal details and submit the necessary documents (may be by scanning).

system to reduce his workload and process the application in a speedy manner.

• Provide a communication platform between the student and the controller.

**IDENTIFYING OBJECTS AND CLASSES**

The concepts of objects and classes are intrinsically linked with each other and form the foundation of object–oriented paradigm.

**Object**

An object is a real-world element in an object–oriented environment that may have a physical or a conceptual existence. Each object has:

Identity that distinguishes it from other objects in the system.

State that determines the characteristic properties of an object as well as the values of the properties that the object holds.

Behavior that represents externally visible activities performed by an object in terms of changes in its state.

Objects can be modelled according to the needs of the application. An object may have a physical existence, like a customer, a car, etc.; or an intangible conceptual existence, like a project, a process, etc.

**Class**

A class represents a collection of objects having same characteristic properties that exhibit common behavior. It gives the blueprint or description of the objects that can be created from it. Creation of an object as a member of a class is called instantiation. Thus, object is an instance of a class.

The constituents of a class are:

A set of attributes for the objects that are to be instantiated from the class. Generally, different objects of a class have some difference in the values of the attributes. Attributes are often referred as class data.

A set of operations that portray the behavior of the objects of the class. Operations are also referred as functions or methods.

**USE CASE:**

The Exam Registration use cases in our system are:

1. Login

2. View exam details

3. Register

4. Acknowledgement

5. Fee Processing

**ACTORS INVOLVED:**

1. Student

2. System DB

**USE-CASE NAME: LOGIN**

The student enters his username and password to login and retrieve the information’.

**USE-CASE NAME: VIEW EXAM DETAILS**

The student view the details about the exam schedule which contains Date,time,etc..

***USE-CASE NAME: REGISTER***

The student should notify the fee details that only the student can pay the correct amount

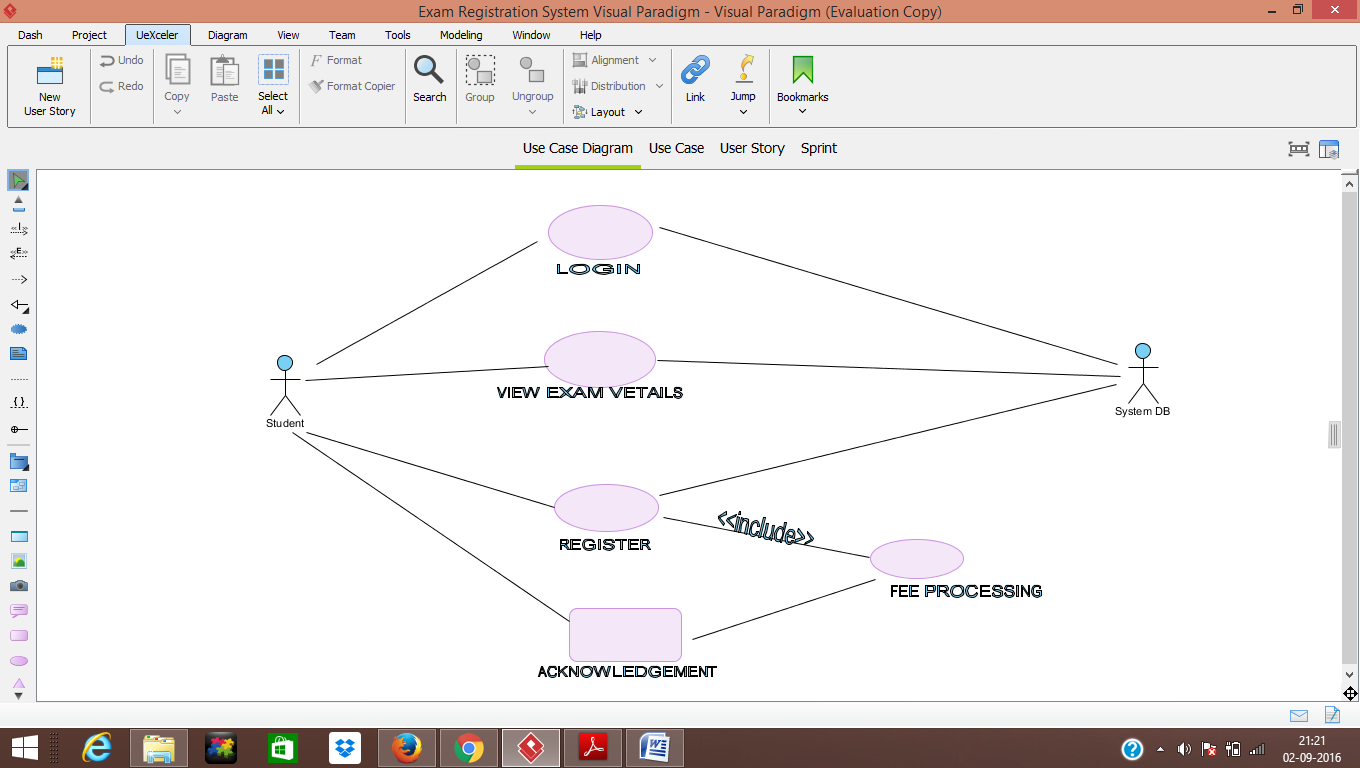
**USE-CASE NAME: ACKNOWLEDGEMENT**

The exam fees should be paid by the student to get the hall ticket from the exam controller.

**USE-CASE NAME: FEE PROCESSING**

*All the details should be viewed by both the student and the controller to verify whether all the entered details are correct.*

**USE CASE DIAGRAM**



**CLASS DIAGRAM:**

**The class diagram, also referred to as object modeling is the main static analysis diagram. The main task of object modeling is to graphically show what each object will do in the problem domain. The problem domain describes the structure and the relationships among objects.**

**The Exam Registration System class diagram consists of four two classes of registration system.**

1. Student\_details

2. Exam\_details3. Register

**1) STUDENT\_DETAILS**

It consists of six attributes and six operations. The attributes id, password, name, age, sex, course. The operations of this class are login(), logout(), conformation(), register(), newfeesdetails().

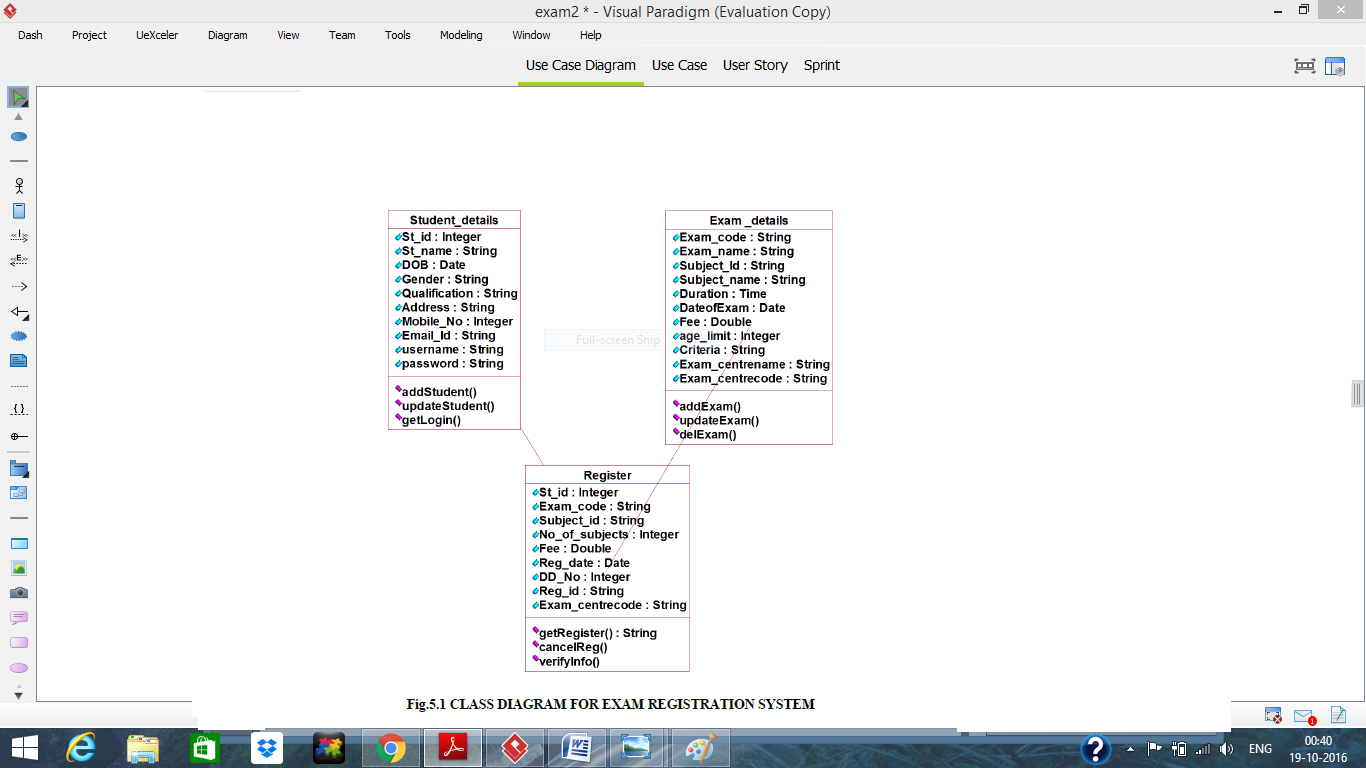
**2) EXAM\_DETAILS**

It consists of four attributes and six methods. The attributes are login(),logout(), feesdetails(), displayfees(), conformation(), examcontroller().

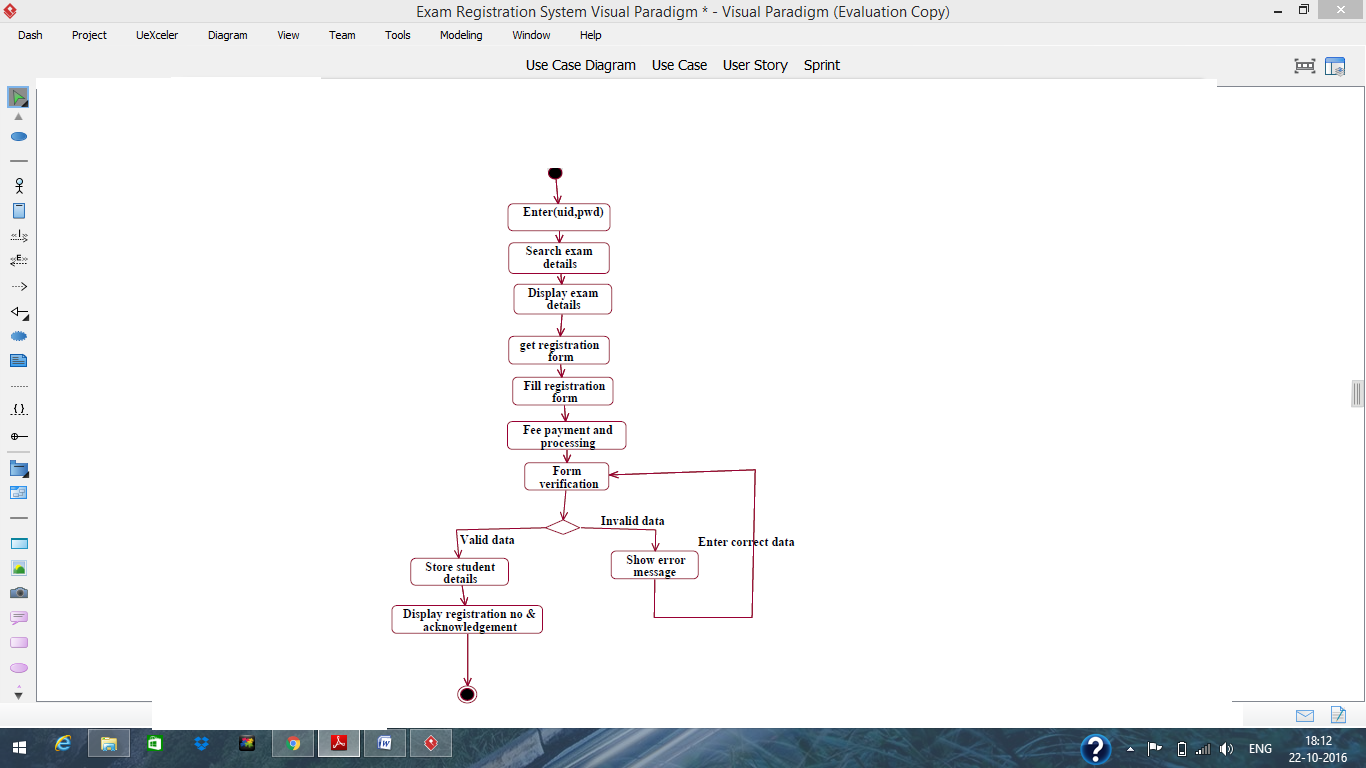
**3) REGISTER**

This class is used to maintain the registered student information such as, subject registered, date of registration and etc,.

**CLASS DIAGRAM FOR EXAM REGISTRATION SYSTEM**



**ACTIVITY DIAGRAM:**

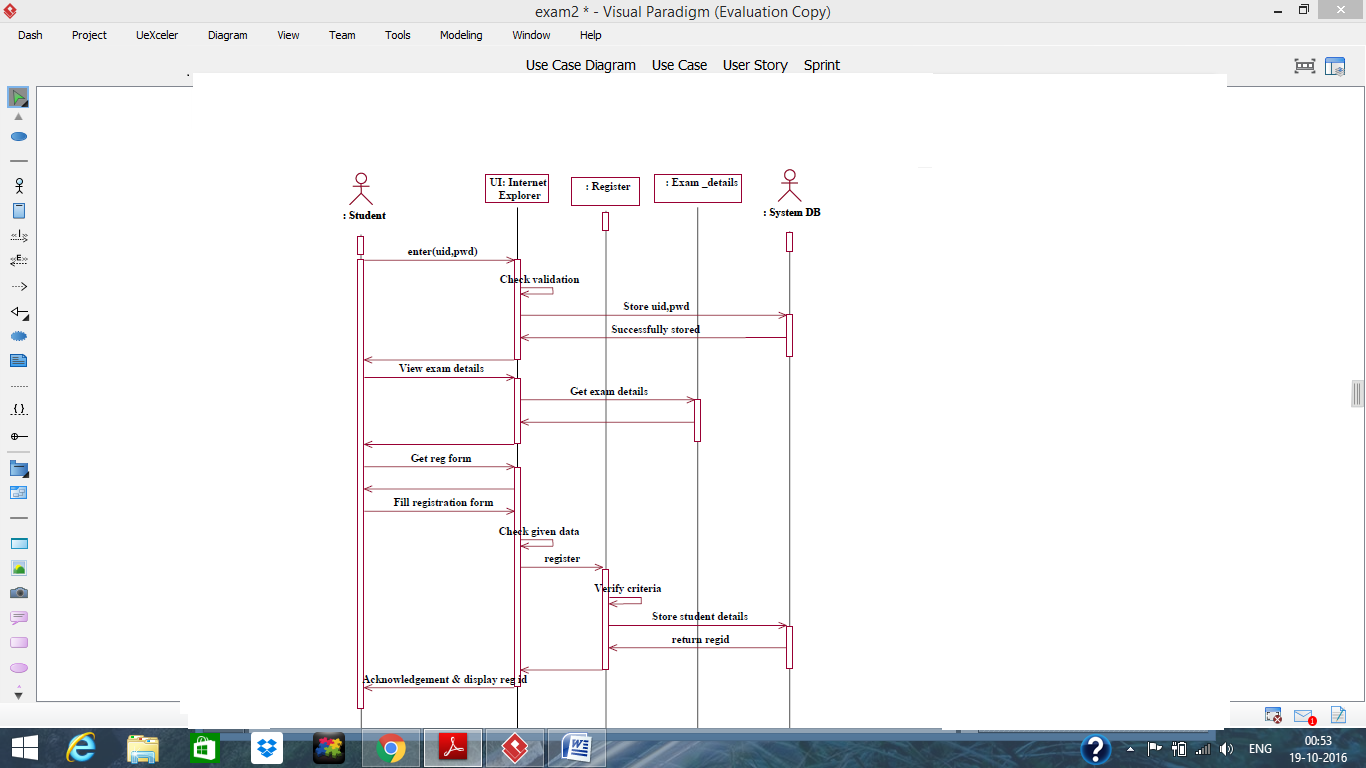


**INTERACTION DIAGRAM:**

A sequence diagram represents the sequence and interactions of a given USE-CASE or scenario. Sequence diagrams can capture most of the information about the system. Most object to object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

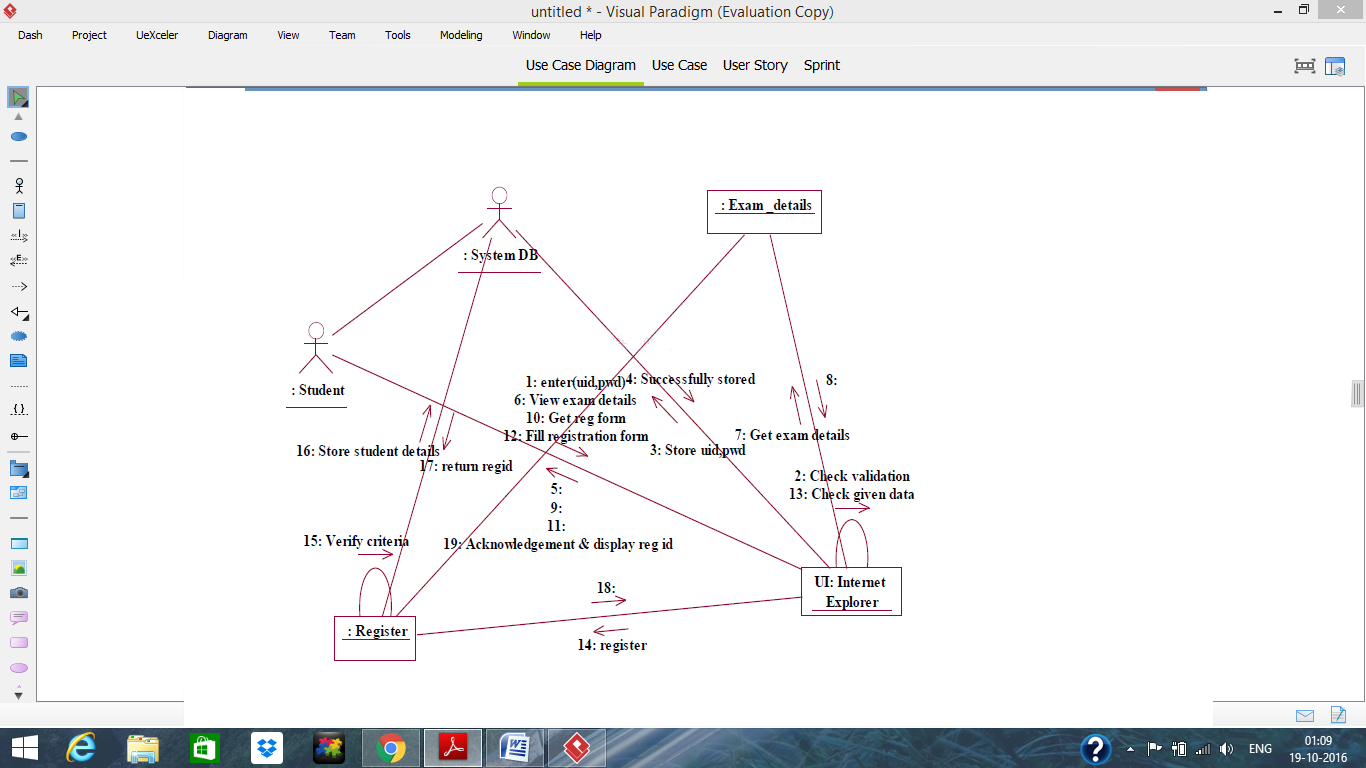
An event also is considered to be any action by an object that sends information. The event line represents a message sent from one object to another, in which the “form” object is requesting an operation be performed by the “to” object. The “to” object performs the operation using a method that the class contains.

It is also represented by the order in which things occur and how the objects in the system send message to one another.



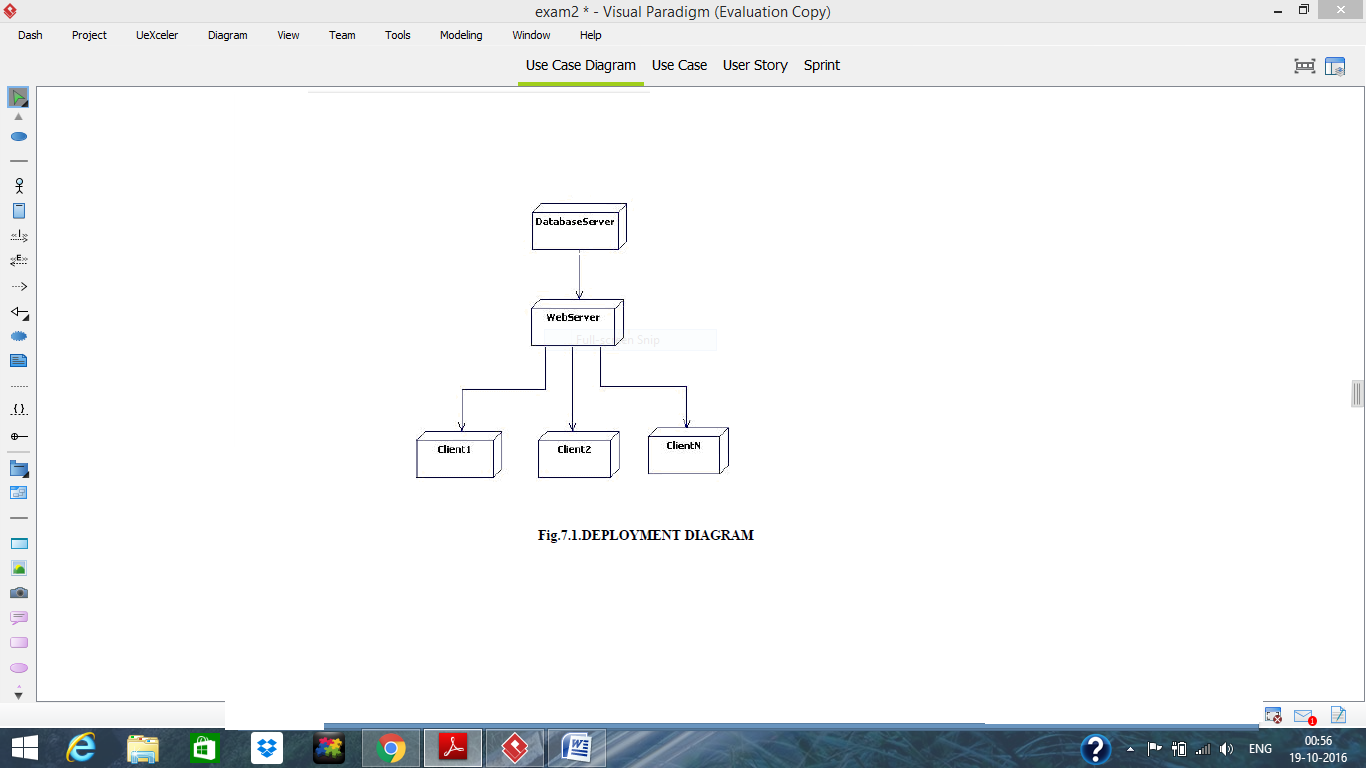
COLLABORATION DIAGRAM

The sequence and collaboration diagram represents that the student enter the information to get the hall ticket and the exam controller issues the hall ticket after verifying the necessary items and this data are stored in the database.



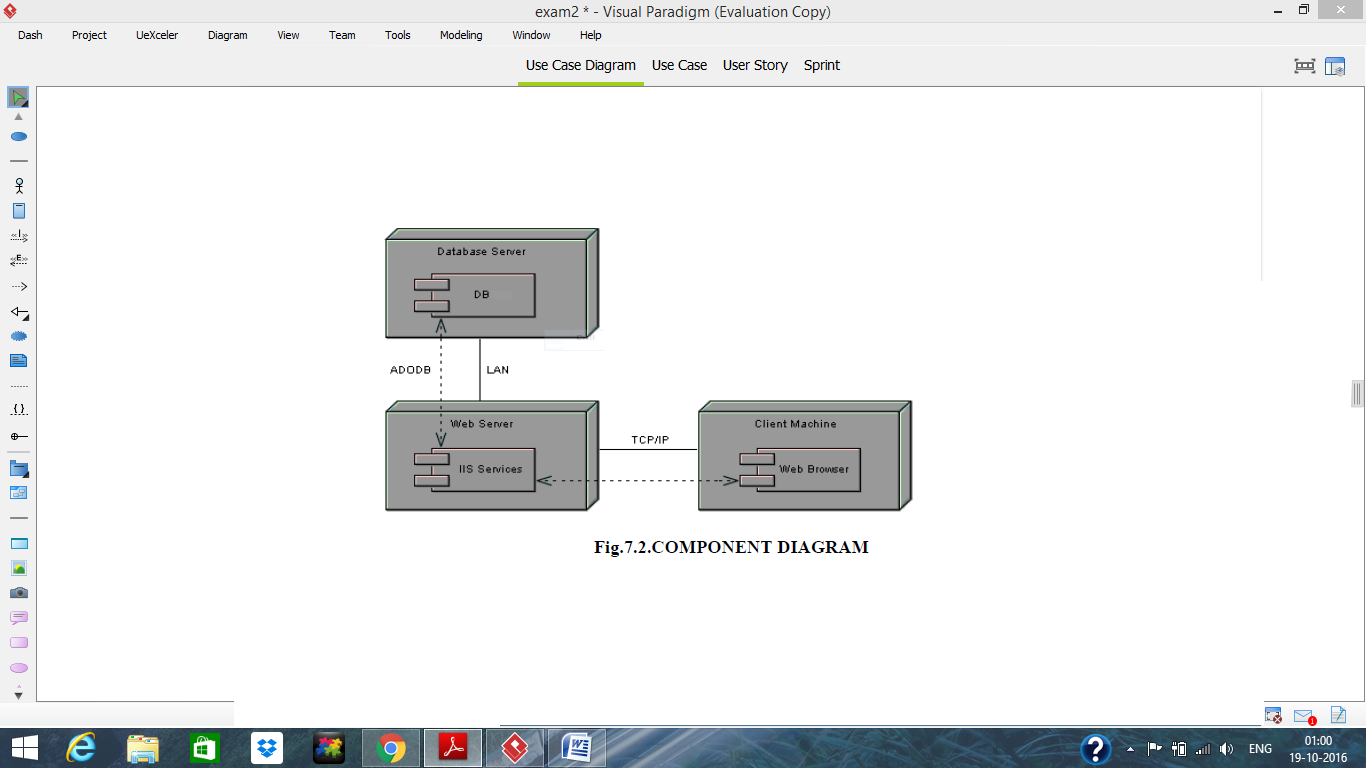
**DEPLOYMENT DIAGRAM**

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.



**COMPONENT DIAGRAM**

Component diagrams are used to visualize the organization and relationships among components in a system.



***GENERATED CODE***

***//Source file: D:\\srmuniversity\\examDetails.java***

*public class examDetails*

*{*

*private string examCode;*

*private string examName;*

*private string subjectId;*

*private string subjectName;*

*private time duration;*

*private date dateOfExam;*

*private double fee;*

*private integer ageLimit;*

*private string criteria;*

*private string examCentreName;*

*private string examCentreCode;*

*public studentDetails theStudentDetails;*

*public register theRegister;*

*/\*\**

*\* @roseuid 515AA57101B5*

*\*/*

*public examDetails()*

*{*

*}*

*/\*\**

*\* @roseuid 515AA448037A*

*\*/*

*public void addExam()*

*{*

*}*

*/\*\**

*\* @roseuid 515AA44F00BB*

*\*/*

*public void updateExam()*

*{*

*}*

*/\*\**

*\* @roseuid 515AA4570280*

*\*/*

*public void delExam()*

*{*

*}*

*}*

*//void examDetails.deleteExam(){*

*//*

*// }*

***//Source file: F:\\Vaish\\Register.java***

*public class Register*

*{*

*private int studid;*

*private String ExamCode;*

*private String subid;*

*private int no.ofSubject;*

*private double fees;*

*private string regid;*

*private String ExamCenterCode;*

*/\*\**

*@roseuid 51342F76033C*

*\*/*

*public Register()*

*{*

*}*

*/\*\**

*@roseuid 51342C88004E*

*\*/*

*public void getRegister()*

*{*

*}*

*/\*\**

*@roseuid 51342C8E0271*

*\*/*

*public void cancelRegister()*

*{*

*}*

*/\*\**

*@roseuid 51342CA20109*

*\*/*

*public void verifyIngormation()*

*{*

*}*

*}*

*//register.register()*

*//register.getregister(){*

*// return null;*

*// }*

*//register.cancelreg(){*

*//*

*// }*

*//register.verifyinfo(){*

*//*

*// }*

***//Source file: F:\\Vaish\\StudentDetails.java***

*public class StudentDetails*

*{*

*private string Studname;*

*private integer Studid;*

*private Date DOB;*

*private String gender;*

*private String qualification;*

*private string Address;*

*private integer mobileno;*

*private string emailid;*

*private string username;*

*private string password;*

*/\*\**

*@roseuid 51342F7602CE*

*\*/*

*public StudentDetails()*

*{*

*}*

*/\*\**

*@roseuid 51342B4901E4*

*\*/*

*public void addStudent()*

*{*

*}*

*/\*\**

*@roseuid 51342B4F03A9*

*\*/*

*public void updateStudent()*

*{*

*}*

*/\*\**

*@roseuid 51342B58029F*

*\*/*

*public void getLogic()*

*{*

*}*

*}*

*/\*\**

*void studentdetails.getlogin(){*

*}*

*studentdetails.studentdetails()*

*void studentdetails.updatestudent(){*

*}*

*void studentdetails.addstudent(){*

*}*

*\*/*

**IMPLEMENTATION OF USER INTERFACE LAYER**

**

***RESULT:***

Thus the mini project for Exam Registration system has been successfully executed and codes are generated.