PROJECT TITLE

A PROJECT REPORT

Submitted By

Mayank Agrawal

(University Roll No 2000290140067)
Rishabh Sharma

(University Roll No 2000290140102)
Pramod Pandey

(University Roll No 2000290140090)
Priyanka Sharma

(University Roll No 2000290140092)

Submitted in partial fulfillment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Mr. Naresh Chandra
Mca Faculty

Submitted to
DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

CERTIFICATE

Certified that Mayank Agrawal (200029014005752), Rishabh Sharma(200029014005787) Pramod Pandey (200029014005775) Priyanka Sharma (200029014005777) have carried out the project work having "IGNITE THE GAMING PLATFORM" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Mayank Agrawal (2000290140067)
Rishabh Sharma (2000290140102)
Pramod Pandey(2000290140090)
Priyanka Sharma(2000290140092)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.
Date:

Mr. Naresh Chandra
MCA Faculty
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Ajay Shrivastava
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, Mr. Naresh Chandra for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot in many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Mayank Agrawal

Rishabh Sharma

Pramod Pandey

Priyanka Sharma

TABLE OF CONTENTS

CHAPTER -1
(INTRODUCTION)

INTRODUCTION

The gaming industry, part of the entertainment industry, comprises three main types of players. From game engines, which help developers build their games. To publishing gaming houses. And gaming consoles. While the prevailing business model for decades has been that of selling the console at cost, and making money on games.

Ignite is going to act as an Information platform(Game search engine**) for the young gamers who want to spend their time in gaming and want to be an esports player.**

Our project will give you following details:-

Game Description.
platforms : Xbox , PlayStation , pc, mobile.
Rating.
Ingame screenshots.

CHAPTER -2
(FEASIBILITY STUDY)

SYSTEM ANALYSIS:

1. Existing System

Existing system is a manual one in which users are maintaining books to store the information like Student Details, Instructor Details, Schedule Details and feedback about students who attempted the exam as per schedule. It is very difficult to maintain historical data.

2. Proposed System

This application is used to find information about the games in the market. The students can sit at individual terminals and login to write the exam in the given duration. The questions have to be given to the students. This application will perform correction, display the result immediately and also store it in the database. This application provides the administrator with a facility to add new GAMES. This application provides the instructor to add questions to the exam, modify questions in the exam in a particular exam. This application takes care of authentication of the administrator, Instructor as Ill as the student.

3. Objective of the System

The objective of the Online GAMING Tool is to provide better information for the users of this system for better results for their maintenance in student GAMING schedule details and grading details.

SYSTEM SPECIFICATION

Hardware Requirements:-

- **Intel or Amd processor**
- **312 MB Ram**
- **Hard Disk minimum 10GB**
- **Other Hardware's like Keyboard ,Mouse**

Software Requirements: -

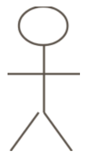
Operating System :	**Windows**
Web-Technology:	**ReactJs NodeJs Mongodb ExpressJs**
Front-End:	**HTML,CSS,JAVASCRIPT**
Back-End:	**NodeJs ExpressJs**

CHAPTER -3
(DESIGN)

UML Diagrams:

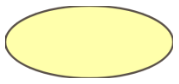
Actor:

A coherent set of roles that users of use cases play when interacting with the use cases.



Use case:

A description of sequence of actions, including variants, that a system performs that yields an observable result of value of an actor.



UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system.

This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representations of the entities that are to be used in the product being developed are needed to be designed.

There are various kinds of methods in software design, they are as follows:

- **Use case Diagram**
- **Sequence Diagram**
- **Collaboration Diagram**
- **Activity Diagram**
- **State chart Diagram**

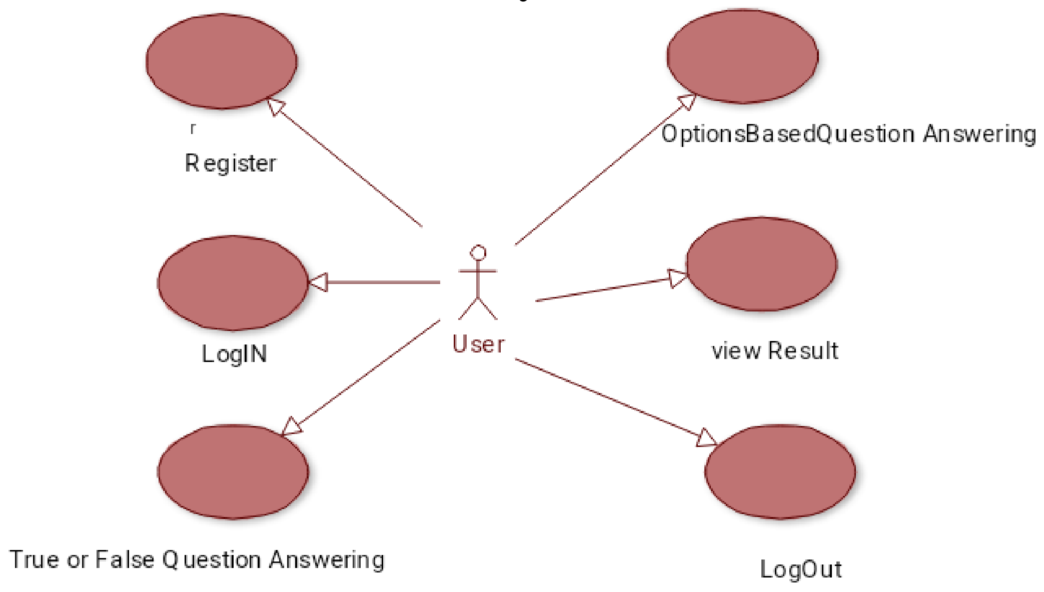
USECASE DIAGRAMS:

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what's called an actor. An actor represents a real-world object

Use case diagram can be useful for getting an overall view of the system and clarifying that can do and more importantly what they can't do. A Use case is a description of a set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name.

Use case diagram consists of use cases and actors and shows the interaction betIen the use case and actors. Primary Actor - Sender, Secondary Actor - Receiver.

- **The purpose is to show the interactions betIen the use case and actor.**
- **To represent the system requirements from user's perspective.**
- **An actor could be the end-user of the system or an external system.**


Register
OptionsBasedQuestion Answering
LogIN
User
view Result
True or False Question Answering
LogOut

SEQUENCE DIAGRAM:

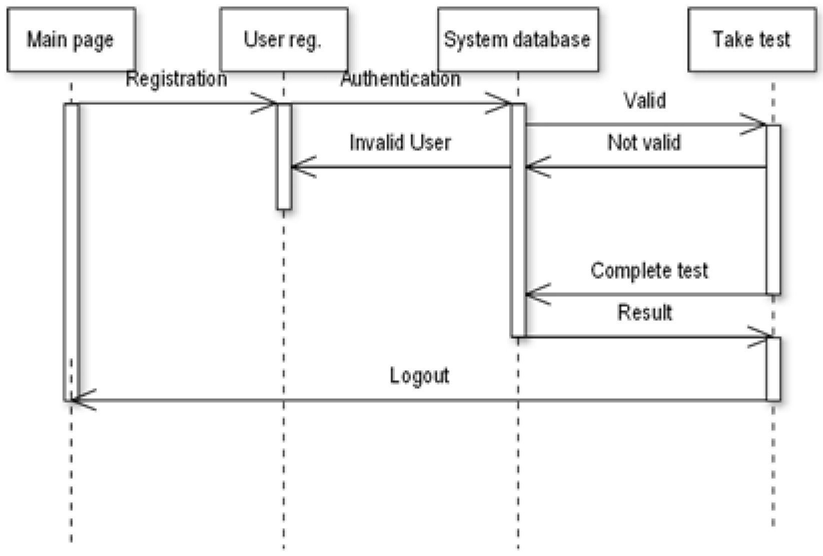
Sequence diagrams and collaboration diagrams are called

INTERACTION DIAGRAMS. An interaction diagram shows an interaction, consisting of a set of objects and their relationship including the messages that may be dispatched among them.

A sequence diagram is an introduction that empathizes the time ordering of messages.

Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages ordered in increasing time along the Y-axis.

Sequence Diagram

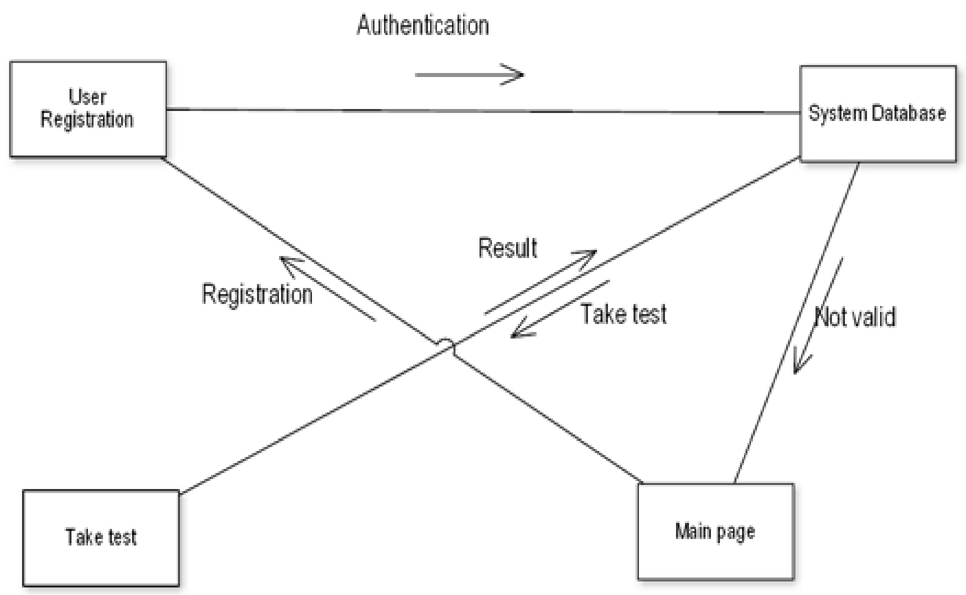


COLLABORATION DIAGRAM:

A collaboration diagram is an introduction diagram that emphasizes the structural organization of the objects that send and receive messages.

Graphically a collaboration diagram is a collection of vertices and arcs.

Collaboration Diagram



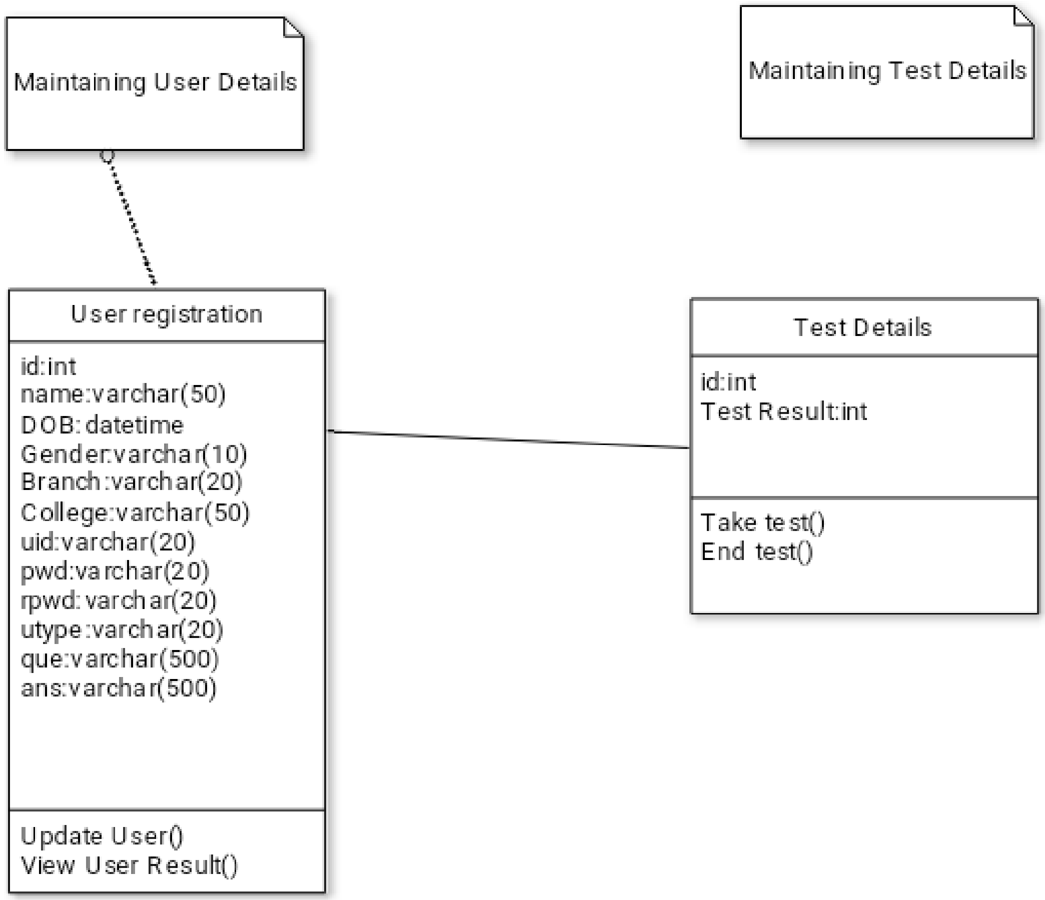
CLASS DIAGRAM:

Class is nothing but a structure that contains both variables and methods. The Class Diagram shows a set of classes, interfaces, and collaborations and their relationships. It shows the dependency betIen the classes that can be used in my system.

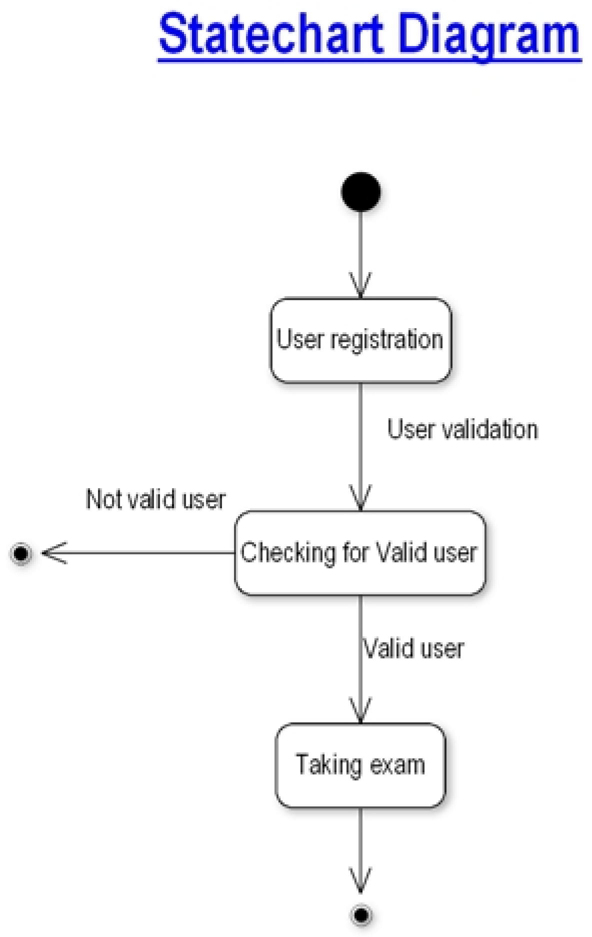
The interactions betIen the modules or classes of my projects are shown below. Each block contains Class Name, Variables and Methods.

CLASS:

A description of a set of objects that share the same attributes operations, relationships, and semantics



STATE CHART DIAGRAM



.

DATA FLOW DIAGRAMS

The DFD takes an input-process-output view of a system i.e. data objects flow into the software, are transformed by

processing elements, and resultant data objects flow out of the software.

Data objects represented by labeled arrows and transformation are represented by circles also called as bubbles. DFD is presented in a hierarchical fashion i.e. the first data flow model represents the system as a whole. Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level.

The DFD enables the software engineer to develop models of the information domain & functional domain at the same time. As the DFD is refined into greater levels of details, the analyst performs an implicit functional decomposition of the system.

At the same time, the DFD refinement results in a corresponding refinement of the data as it moves through the process that embodies the applications.

A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrow represents data objects or object hierarchy.

RULES FOR DFD:

- **Fix the scope of the system by means of context diagrams.**

- **Organize the DFD so that the main sequence of the actions**

- **Reads left to right and top to bottom.**

- **Identify all inputs and outputs.**

- **Identify and label each process internal to the system with Rounded circles.**

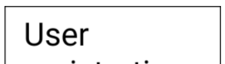
- **A process is required for all the data transformation**

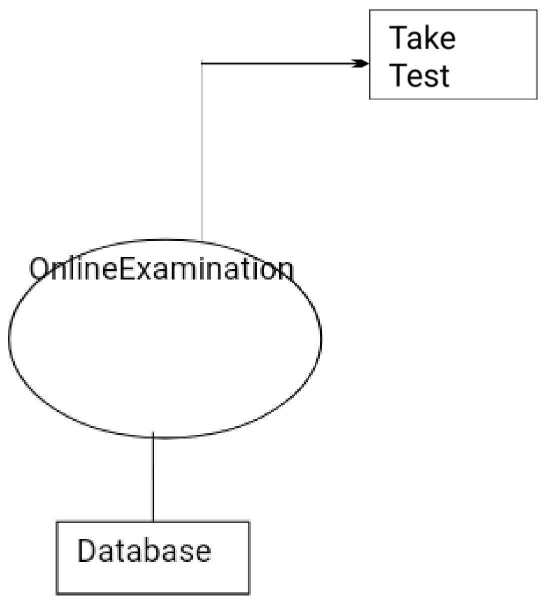
and Transfers. Therefore, never connect a data store to a data Source or the destinations or another data store with just a Data flow arrow.

- **Do not indicate hardware and ignore control information.**
- **Make sure the names of the processes accurately convey everything the process is done.**
- **There must not be unnamed process.**
- **Indicate external source and destinations of the data, with Squares.**
- **Number each occurrence of repeated external entities.**
- **Identify all data flows for each process step, except simple Record retrievals.**
- **Label data flow on each arrow.**
- **Use details flow on each arrow.**
- **Use the details flow arrow to indicate data movements.**

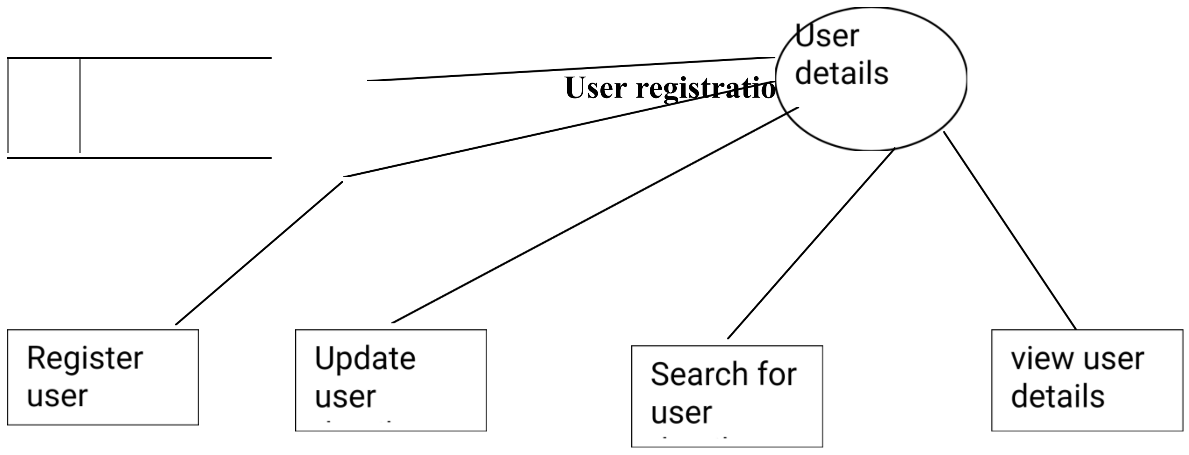
DATAFLOW DIAGRAMS:

Database:

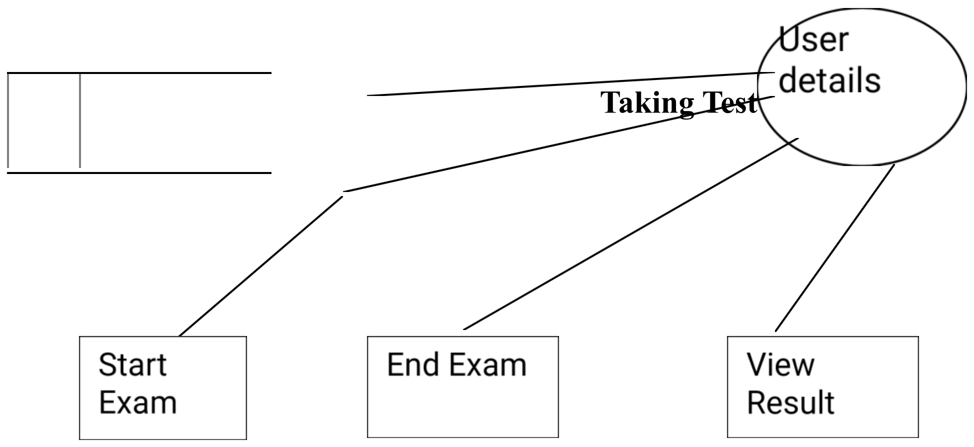




user registration



Taking Test



E-R Diagrams

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 as a way to unify the network and relational database views.

A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database design for the database designer.

The utility of the ER model is:

- It maps Ill to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
- It is simple and easy to understand with a minimum of training.
- Therefore, the model can be used by the database designer to communicate the design to the end user.
- In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

Connectivity and Cardinality

The basic types of connectivity for relations are: one-to-one, one-to-many, and many-to-many. A *one-to-one* (1:1) relationship is when at most one instance of an entity A is associated with one instance of entity B.

For example, "employees in the company are each assigned their own office. For each employee there exists a unique office and for each office there exists a unique employee.

A *one-to-many* (1:N) relationships is when for one instance of entity A, there are zero, one, or many instances of entity B, but for one instance of entity B, there is only one instance of entity A.

An example of a 1:N relationships is a department has many employees each employee is assigned to one department A *many-to-many* (M:N) relationships, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instances of entity B and for one instance of entity B there are zero, one, or many instances of entity A. The connectivity of a relationship describes the mapping of associated

ER Notation

There is no standard for representing data objects in ER diagrams. Each modeling methodology uses its own notation. The original notation used by Chen is widely used in academic texts and journals but rarely seen in either CASE tools or publications by non-academics.

Today, there are a number of notations used among the more common are Bachman, crow's foot, and IDEFIX. All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The symbols used for the basic ER constructs are:

- **Entities are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.**
- **Relationships are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs.**
- **Attributes which are identifiers are underlined. Attribute names should be singular nouns.**
- **Cardinality of many is represented by a line ending in a**

crow's foot. If the crow's foot is omitted, the cardinality is one.

- Existence is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required.

CHAPTER -4
(MODULES)

IGNITE THE GAMING PLATFORM

MODULES:

1: ADMIN MODULE
2. INSTRUCTOR MODULE
3. STUDENT MODULE

1. ADMIN MODULE:

1: REGISTER
2. LOGIN
3. CHANGE PASSWORD OR FORGOT PASSWORD
4. STUDENT -MODIFYING DETAILS
5. DEPARTMENTS-ENTERING/MODIFYING DETAILS
6. INSTRUCTOR DETAILS-MODIFYING DETAILS

1. REGISTER: To be authenticated first have to be registered.

2. LOGIN: The Registered User Can be allowed to view inner details for which he Permitted

3. CHANGE PASSWORD & FORGOT PASSWORD: User has rights to modify his login details & also be informed through mails if he is unable to login.

4. STUDENT -MODIFYING DETAILS: User can be modified to change the status of each User.

5. DEPARTMENTS-ENTERING/MODIFYING DETAILS: New departments adding and old department deletions are spend by this user.

6. INSTRUCTOR DETAILS-MODIFYING DETAILS: According to staff he can add or delete Instructors for specific platforms.

2. INSTRUCTOR MODULE:

i. **REGISTER.**
ii. **LOGIN.**
iii. **CHANGE PASSWORD & FORGOT PASSWORD.**
iv. **ADD QUESTIONS-DEPARTMENTS VERIFYING.**
v. **UPDATE QUESTIONS -DEPARTMENTS VERIFYING.**
vi. **CREATE GAMES.**
vii. **UPDATE GAMES.**
viii. **VIEW EXAM DETAILS- VIEW NO OF REGISTERED STUDENTS**
   1. **VIEW NO OF ATTENDED STUDENTS.**

ix. **EVALUATE QUESTION:MULTIPLE CHOICE**
   1. **TRUE/FALSE.**

1) **REGISTER: To be authenticated first have to be registered.**

2) **LOGIN: The Registered User Can be allowed to view inner details for which he is permitted.**

3) **CHANGE PASSWORD & FORGOT PASSWORD: User has rights to modify his logging details & also be informed through mails if he is unable to login.**

4) **ADD QUESTIONS-DEPARTMENTS VERIFYING: According to flow of questions & Technology he can add questions into the database.**

5) **UPDATE QUESTIONS -DEPARTMENTS VERIFYING: If any corrections in data of questions he can modify them.**

6) **CREATE GAMES: He will be prepared schedule for GAMES periodically.**

7) **UPDATE GAMES: He has rights to modify exam schedule.**

8) VIEW EXAM DETAILS- VIEW NO OF REGISTERED STUDENTS,

a. VIEW NO OF ATTENDED STUDENTS: Can view at attended by students who have registered.

9) EVALUATE QUESTION: MULTIPLE CHOICE

a. TRUE/FALSE: Evaluation of marks based on his initiations when adding questions.

3. STUDENT DETAILS:

a. REGISTER.
b. LOGIN.
c. TAKE EXAM- MULTIPLE CHOICE(Best Option)
d. SEE EXAM RESULTS.
e. LOGOUT.

1) **REGISTER: To be authenticated first have to be registered.**

2) **LOGIN: The Registered User Can be allowed to view inner details for which he is permitted.**

3) **3. TAKE EXAM- MULTIPLE CHOICES, TRUE/FALSE: The registered students are allowed to start the exam.**

4) **SEE EXAM RESULTS: After Completion of exam he can view at his result.**

5) **LOGOUT: After the process of GAMING he turned to Logout page.**

<u>OVERVIEW OF TECHNOLOGIES USED</u>

PHP

PHP: Hypertext Preprocessor, is a widely used, general-purpose scripting language that was originally designed for Web development to produce dynamic Web pages. It can be embedded into HTML and generally runs on a Web server, which needs to be configured to process PHP code and create a Web page content from it. It can be deployed on most Web servers and on almost every operating system and platform free of charge.

PHP was originally created by Rasmus Lerdorf in 1995 and has been in continuous development ever since. The main implementation of PHP is now produced by The PHP Group and serves as the de facto standard for PHP as there is no formal specification.PHP is free software released under the PHP License, which is incompatible with the GNU General Public License (GPL) because of restrictions on the use of the term PHP

PHP has evolved to include a command line interface capability and can also be used in standalone graphical applications.

<u>USAGE</u>

PHP is a general-purpose scripting language that is especially suited for Ib development. PHP generally runs on a Web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic Ib page content. It can also be used for command-line scripting and client-side GUI applications. PHP can be deployed on most Ib servers, many operating systems and platforms, and can be used with many relational database management systems. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

PHP primarily acts as a filter, taking input from a file or stream containing text and/or PHP instructions and outputs

another stream of data; most commonly the output will be HTML. Since PHP 4, the PHP parser compiles input to produce byte code for processing by the Zend Engine, giving improved performance over its interpreter predecessor.

About HTML

HTML, which stands for Hypertext Markup Language, is the predominant markup language for Ib pages. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists etc as Ill as for links, quotes, and other items. It allows images and objects to be embedded and can be used to create interactive forms. It is written in the form of HTML elements consisting of "tags" surrounded by angle brackets within the Ib page content. It can include or can load scripts in languages such as JavaScript which affect the behavior of HTML processors like Ib browsers; and Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both HTML and CSS standards, encmyages the use of CSS over explicit presentational markup.

Hyper Text Markup Language (HTML) is the encoding scheme used to create and format a Ib document. A user need not be an expert programmer to make use of HTML for creating hypertext documents that can be put on the internet.

Most graphical email clients allow the use of a subset of HTML (often ill-defined) to provide formatting and semantic markup not available with plain text. This may include typographic information like colored headings, emphasized and quoted text, inline images and diagrams. Many such clients include both a GUI editor for composing HTML email messages and a rendering engine for displaying them. Use of HTML in email is controversial because of compatibility issues, because it can help disguise phishing attacks, because it can confuse spam filters and because the message size is larger than plain text.

NAMING CONVENTIONS

The most common filename extension for files containing HTML is .html. A common abbreviation of this is .htm, which originated because some early operating systems and file systems, such as DOS and FAT, limited file extensions to three letters.

ABOUT JAVASCRIPT

JavaScript is an object-oriented scripting language used to enable programmatic access to objects within both the client application and other applications. It is primarily used in the form of client-side JavaScript, implemented as an integrated component of the Ib browser, allowing the development of enhanced user interfaces and dynamic Ibsites. JavaScript is a dialect of the ECMAScript standard and is characterized as a dynamic, Iakly typed, prototype-based language with first-class functions. JavaScript was influenced by many languages and was designed to look like Java, but to be easier for non-programmers to work with.

PROTOTYPE-BASED

JavaScript uses prototypes instead of classes for inheritance. It is possible to simulate many class-based features with prototypes in JavaScript.

Functions double as object constructors along with their typical role. Prefixing a function call with new creates a new object and calls that function with its local this keyword bound to that object for that invocation. The constructor's prototype property determines the object used for the new object's internal prototype. JavaScript's built-in constructors, such as Array, also have prototypes that can be modified.

Unlike many object-oriented languages, there is no distinction betIen a function definition and a method

definition. Rather, the distinction occurs during function calling; a function can be called as a method. When a function is called as a method of an object, the function's local this keyword is bound to that object for that invocation.

USAGE

The primary use of JavaScript is to write functions that are embedded in or included from HTML pages and interact with the Document Object Model (DOM) of the page.

Because JavaScript code can run locally in a user's browser (rather than on a remote server) it can respond to user actions quickly, making an application feel more responsive. Furthermore, JavaScript code can detect user actions which HTML alone cannot, such as individual keystrokes. Applications such as Gmail take advantage of this: much of the user-interface logic is written in JavaScript, and JavaScript dispatches requests for information to the server. The wider trend of Ajax programming similarly exploits this strength.

A JavaScript engine (also known as *JavaScript interpreter* or *JavaScript implementation*) is an interpreter that interprets JavaScript soure code and executes the script accordingly. The first JavaScript engine was created by Brendan Eich at Netscape Communications Corporation, for the Netscape Navigator Web browser. A Web browser is by far the most common host environment for JavaScript. Web browsers typically use the public API to create "host objects" responsible for reflecting the DOM into JavaScript.

ABOUT MySQL

MySQL Introduction

There are a large number of database management systems currently available, some commercial and some free.

Some of them: Oracle, Microsoft Access, MySQL and PostgreSQL.

These database systems are powerful, feature-rich software, capable of organizing and searching millions of records at

very high speeds.

Every Database is composed of one or more tables. These Tables, which structure data into rows and columns, impose organization on the data.

To make it easy to identify a specific record, therefore, it becomes necessary standing Relationships and Foreign Keys (RDBMS).

You already know that a single database can hold multiple tables.
In a Relational database management system (RDBMS).

These tables can be linked to each other by one or more common fields, called foreign keys.

FEASIBILITY STUDY:

Feasibility study is conducted once the problem is clearly understood. Feasibility study is a high level capsule version of the entire system analysis and design process. The objective is to determine quickly at a minimum expense how to solve a problem. The purpose of feasibility is not to solve the problem but to determine if the problem is worth solving.

The system has been tested for feasibility in the following points.

1. **Technical Feasibility**
2. **Economical Feasibility**
3. **Operational Feasibility.**

1. Technical Feasibility

The project entitles "IGNITE THE GAMING PLATFORM" is technically feasibility because of the below mentioned feature. The project was developed in Java which Graphical User Interface. All these make Java an appropriate language for this project. Thus the existing software Java is a powerful language.

2. Economical Feasibility

The computerized system will help automate the selection leading the profits and details of the organization. With this software, the machine and manpower utilization are expected to go up by 80-90% approximately. The costs incurred of not creating the system are set to be great, because precious time can be wanted by manually.

3. Operational Feasibility

In this project, the management will know the details of each project where he may be presented and the data will be

maintained as decentralized and if any inquires for that particular contract can be known as per their requirements and necessities.

CHAPTER -5
(CODING)

Implementation:

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system is giving confidence on the new system for the users that it will work efficiently and effectively.
The system can be implemented only after thorough testing is done and if it is found to work according to the specification.

It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods apart from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The more complex the system being implemented, the more involved will be the system analysis and design effort required for implementation.

The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

TESTING:

The testing phase is an important part of software development. It is the puterized system will help automate

the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. Software testing is carried out in three steps:

1. The first includes unit testing, where in each module is tested to provide its correctness, validity and also determine any missing operations and to verify whether the objectives have been met. Errors are noted down and corrected immediately. Unit testing is an important and major part of the project. So errors are rectified easily in particular module and program clarity is increased. In this project the entire system is divided into several modules and is developed individually. So unit testing is conducted to individual modules.

2 The second step includes Integration testing. It need not be the case, the software whose modules when run individually and showing perfect results, will also show perfect results when run as a whole. The individual modules are clipped under this major module and tested again and verified the results. This is due to poor interfacing, which may result in data being lost across an interface. A module can have inadvertent, adverse effect on any other or on the global data structures, causing serious problems.

3. The final step involves validation and testing which determines which the software functions as the user expected. Here also some modifications Ire. In the completion of the project it is satisfied fully by the end user.

Maintenance and environment:

AS the number of computer based systems, grieve libraries of computer software began to expand. In house developed projects produced tones of thousand soft program smyce statements. Software products purchased from the outside added hundreds of thousands of new statements. A dark cloud appeared on the horizon. All of these programs, all of those smyce statements-had to be corrected when false Ire detected, modified as user requirements changed, or

adapted to new hardware that was purchased. These activities Ire collectively called software Maintenance.

The maintenance phase focuses on change that is associated with error correction, adaptations required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements. Fmy types of changes are encountered during the maintenance phase.

Correction
Adaptation
Enhancement
Prevention

Correction:

Even with the best quality assurance activities is lightly that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

Maintenance is a set of software Engineering activities that occur after software has been delivered to the customer and put into operation. Software configuration management is a set of tracking and control activities that began when a software project begins and terminates only when the software is taken out of the operation.
I may define maintenance by describing fmy activities that are undertaken after a program is released for use:

Corrective Maintenance
Adaptive Maintenance
Perfective Maintenance or Enhancement
Preventive Maintenance or reengineering

Only about 20 percent of all maintenance work are spent "fixing mistakes". The remaining 80 percent are spent adapting existing systems to changes in their external environment, making enhancements

requested by users, and reengineering an application for use.

ADAPTATION:

Over time, the original environment (E>G., CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate change to its external environment.

ENHANCEMENT:

As software is used, the customer/user will recognize additional functions that will provide benefit. Perceptive maintenance extends the software beyond its original function requirements.

PREVENTION:

Computer software deteriorates due to change, and because of this, preventive maintenance, often called software re engineering, must be conducted to enable the software to serve the needs of its end users. In essence, preventive maintenance makes changes to computer programs so that they can be more easily corrected, adapted, and enhanced. Software configuration management (SCM) is an umbrella activity that is applied throughout the software process. SCM activities are developed to

<u>SOFTWARE METHODOLOGY</u>

The software methodology followed in this project includes

the object-oriented methodology and the application system development methodologies. The description of these methodologies is given below.

Application System Development – A Life cycle Approach

Although there are a growing number of applications (such as decision support systems) that should be developed using an experimental process strategy such as prototyping, a significant amount of new development work continues to involve major operational applications of broad scope. The application systems are large highly structured. User task comprehension and developer task proficiency is usually high. These factors suggest a linear or iterative assurance strategy. The most common method for this stage class of problems is a system development life cycle model in which each stage of development is Ill defined and has straightforward requirements for deliverables, feedback and sign off. The system development life cycle is described in detail since it continues to be an appropriate methodology for a significant part of new development work.

The basic idea of the system development life cycle is that there is a Well-defined process by which an application is conceived and developed and implemented. The life cycle gives structure to a creative process. In order to manage and control the development effort, it is necessary to know what should have been done, what has been done, and what has yet to be accomplished. The phrases in the system development life cycle provide a basis for management and control because they define segments of the flow of work, which can be identified for managerial purposes and specifies the documents or other deliverables to be produced in each phase.

The phases in the life cycle for information system development are described differently by different writers, but the differences are primarily in the amount of necessity and manner of categorization. There is a general agreement

on the flow of development steps and the necessity for control procedures at each stage.

The information system development cycle for an application consists of three major stages.

1) Definition.
2) Development.
3) Installation and operation.

The first stage of the process defines the information requirements for a feasible cost effective system. The requirements are then translated into a physical system of forms, procedures, programs etc. by the system design, computer programming and procedure development. The resulting system is test and put into operation. No system is perfect so there is always a need for maintenance changes. To complete the cycle, there should be a post audit of the system to evaluate how Ill it performs and how Ill it meets the cost and performance specifications. The stages of definition, development and installation and operation can therefore be divided into smaller steps or phrases as follows.

<u>Definition</u>

Proposed definition: preparation of request for proposed applications.
Feasibility assessment: evaluation of feasibility and cost benefit of proposed system.
Information requirement analysis: determination of information needed.

<u>Design</u>

Conceptual design: User-oriented design of application

development.
Physical system design: Detailed design of flows and processes in applications processing system and preparation of program specification.

Development

Program development: coding and testing of computer programs.
Procedure development: design of procedures and preparation of user instructions.

Installation and operation

Conversion: final system test and conversion.
Operation and maintenance: Month to month operation and maintenance

Post audit: Evaluation of development process, application system and results of use at the completion of each phase, formal approval sign-off is required from the users as Ill as from the manager of the project development.

CHAPTER -6
(TESTING)

Testing is a process of executing a program with the indent of finding an error. Testing is a crucial element of software quality assurance and presents ultimate review of

specification, design and coding.

System Testing is an important phase. Testing represents an interesting anomaly for the software. Thus a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

A good test case is one that has a high probability of finding an undiscovered error. A successful test is one that uncovers an undiscovered error.

Testing Objectives:

1. **Testing is a process of executing a program with the intent of finding an error**
2. **A good test case is one that has a probability of finding an as yet undiscovered error**
3. **A successful test is one that uncovers an undiscovered error**

Testing Principles

1. **All tests should be traceable to end user requirements**
2. **Tests should be planned long before testing begins**
3. **Testing should begin on a small scale and progress towards testing in large**
4. **Exhaustive testing is not possible**
5. **To be most effective testing should be conducted by an independent third party**

The primary objective for test case design is to derive a set of tests that has the highest livelihood for uncovering defects in software. To accomplish this objective two different categories of test case design techniques are used.

- **White box testing.**
- **Black box testing.**

White-box testing:

White box testing focus on the program control structure. Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.

Black-box testing:

Black box testing is designed to validate functional requirements without regard to the internal workings of a program. Black box testing mainly focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides through test coverage. Incorrect and missing functions, interface errors, errors in data structures, error in functional logic are the errors falling in this category.

Testing strategies:

A strategy for software testing must accommodate low-level tests that are necessary to verify that all small smyce code segments has been correctly implemented as Ill as high-level tests that validate major system functions against customer requirements.

Testing fundamentals:

Testing is a process of executing a program with the intent of finding errors. A good test case is one that has a high probability of finding an undiscovered error. If testing is conducted successfully it uncovers the errors in the software. Testing cannot show the absence of defects, it can only show that software defects present.

Testing Information flow:

Information flow for testing flows the pattern. Two class of input provided to test the process. The software configuration includes a software requirements specification, a design specification and source code.

Test configuration includes test plan and test cases and test

tools. Tests are conducted and all the results are evaluated. That is test results are compared with expected results. When erroneous data are uncovered, an error is implied and debugging commences.

Unit testing:

Unit testing is essential for the verification of the code produced during the coding phase and hence the goal is to test the internal logic of the modules. Using the detailed design description as a guide, important paths are tested to uncover errors within the boundary of the modules. These tests Ire carried out during the programming stage itself

Integration testing:

Integration testing focuses on unit tested modules and build the program structure that is dictated by the design phase.

System testing:

System testing tests the integration of each module in the system. It also tests to find discrepancies between the system and its original objective, current specification and system documentation.

The primary concern is the compatibility of individual modules.

Entire system is working properly or not will be tested here, and specified path ODBC connection will correct or not, and giving output or not are tested here these verifications and validations are done by giving input values to the system and by comparing with expected output.

Top- down testing implementing here.

Acceptance Testing:

This testing is done to verify the readiness of the system for the implementation. Acceptance testing begins when the system is complete. Its purpose is to provide the end user with the confidence that the system is ready for use. It

involves planning and execution of functional tests, performance tests and stress tests in order to demonstrate that the implemented system satisfies its requirements.

Tools to special importance during acceptance testing include: Test coverage Analyzer – records the control paths folloId for each test case.

Timing Analyzer – also called a profiler, reports the time spent in various regions of the code are areas to concentrate on to improve system performance.

Coding standards – static analyzers and standard checkers are used to inspect code for deviations from standards and guidelines.

Test Cases:

Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.

Using White-Box testing methods, the software engineer can drive test cases that

- **Guarantee that logical decisions on their true and false sides.**
- **Exercise all logical decisions on their true and false sides.**
- **Execute all loops at their boundaries and with in their operational bounds.**
- **Exercise internal data structure to assure their validity.**

CHAPTER -7
(CONCLUSION)

CONCLUSION:

The package was designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project.

- **Automation of the entire system improves the efficiency**
- **It provides a friendly graphical user interface which proves to be better when compared to the existing system.**
- **It gives appropriate access to the authorized users depending on their permissions.**
- **It effectively overcomes the delay in communications.**
- **Updating of information becomes so easier.**
- **System security, data security and reliability are the striking features.**
- **The System has adequate scope for modification in the future if it is necessary.**

CHAPTER -8
(ENHANCEMENTS)

FUTURE ENHANCEMENTS:

This application avoids the manual work and the problems concern with it. It is an easy way to obtain the information regarding the different scheduled GAMINGs information that is currently issued.

Well I and my team members have worked hard in order to present an improved Website better than the existing one's regarding the information about the various activities.

Still, I found out that the project can be done in a better way. Primarily, when I request information about a particular schedules it just shows the exam date and platform. So, after getting the information I can get access to the online exam

CHAPTER -9
(BIBILIOGRAPHY)

BIBLIOGRAPHY

The following Youtube Channels Were referred during the analysis and execution phase of the project

✓ Videos Referred:

- **https://www.youtube.com/roadsidecoder**
- **https://www.youtube.com/c/DevEd**
- **https://www.youtube.com/c/Telusko**
- **https://www.youtube.com/c/CodeWithHarry**