**INDEX**

|  |  |
| --- | --- |
| **SL. No** | **CONTENTS** |
| **1.** | **INTRODUCTION** |
| **2.** | **ANALYSIS** |
| **2.1** | *SYSTEM ANALYSIS* |
| **2.2** | *SYSTEM SPECIFICATIONS* |
| **3.** | **DESIGN APPROACH** |
| **3.1** | *INTRODUCTION TO DESIGN* |
| **3.2** | *UML DIAGRAMS* |
| **3.3** | *DATA FLOW DIAGRAMS* |
| **3.4** | *E-R DIAGRAMS* |
| **4.** | **PROJECT MODULES** |
| **5.** | **IMPLEMENTATION** |
| **5.1** | *CONCEPTS AND TECHNIQUES* |
| **5.2** | *TESTING* |
| **5.2.1** | *TEST CASES* |
| **6.** | **OUTPUT SCREENS** |
| **7.** | **CONCLUSIONS** |
| **8.** | **FUTURE ENHANCEMENTS** |
| **9.** | **BIBILIOGRAPHY** |

**INTRODUCTION**

On-line examinations contents providers to focus on creating effective assessment questions and focusing on exam’s feedback delivery to students. In the paper we present techniques that are pertinent to the elements of assessment process: answers submission, computerized grading, and feedback after submission.

As the modern organizations are automated and computers are working as per the instructions, it becomes essential for the coordination of human beings, commodity and computers in a modern organization.

The administrators, instructor, Students who are attending for online examination can communicate with the system through this project, thus facilitating effective implementation and monitoring of various activities of Online Examinations like conducting Exams as per scheduled basis and delivering result to that particular use or student. And the details of students who attempted Online Examination are maintained at administrator.

**ANALYSIS**

**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 feedbacks about students who attempted exam as per schedule. It is very difficult to maintain historical data.

DISADVANTAGES:

The following drawbacks of existing system emphasize the need for computerization:

1. A lot of copies of question papers must be made

2. A lot of correction work hence delay in giving the results

3. A lot of tabulation work for each subject results

2. **Proposed System**

This application is used to conduct online examination. The students can sit at individual terminals and login to write the exam in the given duration. The questions must be given to the students. This application will perform correction, display the result immediately and store it in database. This application provides the administrator with a facility to add new exams. This application provides the instructor 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 well as the student.

3. **Objective of the System**

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

**System Specifications**

Hardware Requirements: -

• Pentium-IV(Processor).

• 256 MB Ram

• 512 KB Cache Memory

• Hard disk 10 GB

• Microsoft Compatible 101 or more Keyboard

Software Requirements: -

• Operating System: Windows

• Web-Technology: PHP

• Front-End: HTML, CSS,JAVASCRIPT

• Back-End: MySQL

• Web Server: Apache SERVER.

**DESIGN**

**INTRODUCTION:**

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analysed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

**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 Modelling 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 representation of the entities that are to be used in the product being developed need 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 chat Diagram

**USECASE DIAGRAMS:**

Use case diagrams model behaviour within a system and helps the developers understand of what the user require. The stick man represents what’s called an actor.

Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can’t do.

Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

* The purpose is to show the interactions between 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.

**USECASE DIAGRAM:**

A Use case is a description of set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship. It is an association between the use cases and actors. An actor represents a real-world object. Primary Actor – Sender, Secondary Actor Receiver.



**SEQUENCE DIAGRAM:**

Sequence diagram and collaboration diagram are called INTERACTION DIAGRAMS. An interaction diagram shows an interaction, consisting of 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.

Diagram

Description automatically generated

**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.

Diagram

Description automatically generated

**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 relating ships. There is most common diagram in modeling the object-oriented systems and are used to give the static view of a system. It shows the dependency between the classes that can be used in our system.

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

**CLASS:**

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



**State Chart Diagram**

Diagram

Description automatically generated

.

**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 labelled 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. 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 embody 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 labelled 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 sources 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

Take Test

Database

**user registration**

**User registration**

view user details

Update user details

Register user

Search for user details

**Taking Test**

**Taking Test**

Start Exam

End Exam

View Result

**E-R Diagrams:**

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 [Chen76] to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represents 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 well 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 a specific database management software.

**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 academics 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 notation used in this document is from Martin. The symbols used for the basic ER constructs are:

* **entities** are represented by labelled 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**, when included, are listed inside the entity rectangle. 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. Optional existence is shown by placing a circle next to the entity that is optional.

**MODULES**

**1: ADMIN MODULE**

**2.INSTRUCTOR MODULE**

**3.STUDENT MODULE**

**1.ADMIN MODULE:**

**1.REGISTER**

**2.LOGIN**

**3.CHANGE PASSWORD&FORGOTPASSWORD**

**4.STUDENT -MODIFING 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 -MODIFING DETAILS:** User can be modified to change status of each User.

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

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

**2. INSTRUCTOR MODULE:**

**1.REGISTER**

**2. LOGIN**

**3.CHANGE PASSWORD&FORGOT PASSWORD**

**4.ADD QUESTIONS-DEPARTMENTS VERIFING.**

**5.UPDATE QUESTIONS -DEPARTMENTS VERIFING**

**6.CREATE EXAMS**

**7.UPDATE EXAMS**

**8.VIEW EXAM DETAILS- VIEW NO OF REGISTERED STUDENTS**

**VIEW NO OF ATTENDED STUDENTS**

**9.EVALUATE QUESTION: MULTIPLE CHOICE**

**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 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 VERIFING:** According to flow of questions & Technology he can add questions into the database.

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

**6.CREATE EXAMS:** He will be prepared schedule for exams periodically.

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

**8.VIEW EXAM DETAILS-** VIEW NO OF REGISTERED STUDENTS,

VIEW NO OF ATTENDED STUDENTS: Can view at attended students who has registered.

**9.EVALUATE QUESTION:** MULTIPLE CHOICE

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

**3. STUDENT DETAILS:**

**1.REGISTER**

**2.LOGIN**

**3.TAKE EXAM- MULTIPLE CHOICE**

**TRUE/FALSE**

**4. SEE EXAM RESULTS**

**5.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

Permitted

**3.TAKE EXAM- MULTIPLE CHOICE, TRUE/FALSE:** The registered student 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 examination, he turned to Logout page.

**DATABASE TABLES:**

**USER REG TABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | NULL/NOTNULL | TYPE | KEY |
| ID | NOTNULL | INT | PRIMARYKEY |
| NAME | NULL | VARCHAR(50) |  |
| DOB | NULL | DATETIME |  |
| GENDER | NULL | VARCHAR(10) |  |
| BRANCH | NULL | VARCHAR(20) |  |
| CPFNO | NULL | VARCHAR(50) |  |
| UID | NULL | VARCHAR(50) |  |
| PWD | NULL | VARCHAR(20) |  |
| RPWD | NULL | VARCHAR(20) |  |
| UTYPE | NULL | VARCHAR(20) |  |
| QUE | NULL | VARCHAR(500) |  |
| ANS | NULL | VARCHAR(500) |  |

**True/False Based Question Table**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | NULL/NOTNULL | TYPE | KEY |
| ID | NOTNULL | INT | PRIMARYKEY |
| QUE | NULL | VARCHAR(500) |  |
| AW | NULL | VARCHAR(500) |  |

**True/False Based Answer Table**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | NULL/NOTNULL | TYPE | KEY |
| ID | NOTNULL | INT | FOREIGNKEY |
| AW | NULL | VARCHAR(500) |  |

**Options Based Question Table**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | NULL/NOTNULL | TYPE | KEY |
| QID | NOTNULL | INT | PRIMARYKEY |
| QN | NULL | VARCHAR(500) |  |
| OPTIONS1 | NULL | VARCHAR(100) |  |
| OPTIONS2 | NULL | VARCHAR(100) |  |
| ANSWER | NULL | VARCHAR(100) |  |

**Options Based Answers**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | NULL/NOTNULL | TYPE | KEY |
| QID | NOTNULL | INT | FOREIGNKEY |
| ANSWER | NULL | VARCHAR(10) |  |

**All Student Marks**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | NULL/NOTNULL | TYPE | KEY |
| ID | NULL | INT |  |
| MARKS | NULL | INT |  |

**Exam Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | NULL/NOTNULL | TYPE | KEY |
| ENAME | NULL | VARCHAR(30) |  |
| EDATE | NULL | DATETIME |  |

**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 and in 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 changeover and an evaluation of change over methods a part 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 systems analysis and design effort required just 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**

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 as undiscovered error. A successful test is one that uncovers an as 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 a 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. They are

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.

**Block-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 source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

**Testing fundamentals:**

Testing is a process of executing program with the intent of finding error. A good test case is one that has 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 with in the boundary of the modules. These tests were carried out during the programming stage itself. All units of ViennaSQL were successfully tested.

**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 it’s 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 followed 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.

The test case specification for system testing has to be submitted for review before system testing commences.

**OUTPUT SCREENS**

Graphical user interface, website

Description automatically generated

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, website

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**CONCLUSION**

The package was designed in such a way that future modifications can 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 future if it is necessary.

**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 examinations information that are

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

The enhancement that we can add the searching option. We can directly search to the particular student details from this site.

**BIBLIOGRAPHY**

The following books were referred during the analysis and execution phase of the project

* **Books Referred:**

• BEGINNING PHP 5 ---DAVE MERCER

• BLACK BOOK HTML ---WILEY DREAMTECH

• PHP AND MYSQL WEB DEVELOPMENT --- LUKEWELLING, LAURA

• MICROSOFT SQL SERVER-2000 ---RANKIN, PAUL & JENSEN

• SQL SERVER-2000 ---DUSAN PETKOVIC

• PHP IN A NUTSHELL --- PAUL HUDSON