1 A).Aim: Requirement analysis and SRS forCourse Registration system.

Requirements:

Hardware Requirements: PC with 300 megahertz or higher processor clock speed recommended; 233 MHz

- minimum required. 128 megabytes (MB) of RAM or higher recommended (64 MB minimum
- supported)
- 1.5 gigabytes (GB) of available hard disk space
- CD ROM or DVD Drive
- Keyboard and Mouse(compatible pointing device).

Software Requirements:

- Rational Rose, Windows XP,

Theory:

An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.

The SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an eCommerce Web site, and so on) must provide, as well as states any required constraints by which the system must abide. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

It's important to note that an SRS contains functional and nonfunctional requirements only; it doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements to be.

A well-designed, well-written SRS accomplishes four major goals:

It provides feedback to the customer. An SRS is the customer's assurance that the

- development organization understands the issues or problems to be solved and the software behavior necessary to address those problems. Therefore, the SRS should be written in natural language (versus a formal language, explained later in this article), in an unambiguous manner that may also include charts, tables, data flow diagrams, decision tables, and so on.
- It decomposes the problem into component parts. The simple act of writing down software requirements in a well-designed format organizes information, places borders around the problem, solidifies ideas, and helps break down the problem into its component parts in an orderly fashion.
- It serves as an input to the design specification. As mentioned previously, the SRS serves as the parent document to subsequent documents, such as the software design specification and statement of work. Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised.
- It serves as a product validation check. The SRS also serves as the parent document for testing and validation strategies that will be applied to the requirements for verification.

SRSs are typically developed during the first stages of "Requirements Development," which is the initial product development phase in which information is gathered about what requirements are needed--and not. This information-gathering stage can include onsite visits, questionnaires, surveys, interviews, and perhaps a return-on-investment (ROI) analysis or needs analysis of the customer or client's current business environment. The actual specification, then, is written after the requirements have been gathered and analyzed.

SRS should address the following The basic issues that the SRS shall address are the following:

a) **Functionality.** What is the software supposed to do?

b) **External interfaces.** How does the software interact with people, the system‘s hardware, other hardware, and other software?

c) **Performance**. What is the speed, availability, response time, recovery time of various software functions, etc.?

d) **Attributes.** What are the portability, correctness, maintainability, security, etc. considerations?

e) Design constraints imposed on an implementation. Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

Chracteristics of a good SRS An SRS should be

a) Correct

b) Unambiguous

c) Complete

d) Consistent

e) Ranked for importance and/or stability

f) Verifiable

g) Modifiable

h) Traceable

Correct - This is like motherhood and apple pie. Of course you want the specification to be correct. No one writes a specification that they know is incorrect. We like to say - "Correct and Ever Correcting." The discipline is keeping the specification up to date when you find things that are not correct.

Unambiguous - An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. Again, easier said than done. Spending time on this area prior to releasing the SRS can be a waste of time. But as you find ambiguities - fix them.

Complete - A simple judge of this is that is should be all that is needed by the software designers to create the software.

Consistent - The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another.

Ranked for Importance - Very often a new system has requirements that are really marketing wish lists. Some may not be achievable. It is useful provide this information in the SRS.

Verifiable - Don't put in requirements like - "It should provide the user a fast response." Another of my favorites is - "The system should never crash." Instead, provide a quantitative requirement like: "Every key stroke should provide a user response within 100 milliseconds."

Modifiable - Having the same requirement in more than one place may not be wrong - but tends to make the document not maintainable.

Traceable - Often, this is not important in a non-politicized environment. However, in most organizations, it is sometimes useful to connect the requirements in the SRS to a higher level document. Why do we need this requirement?

A sample of basic SRS Outline

1. Introduction

1.1 Purpose

1.2 Document conventions

1.3 Intended audience

1.4 Additional information

1.5 Contact information/SRS team members

1.6 References

2. Overall Description

2.1 Product perspective

2.2 Product functions

2.3 User classes and characteristics

2.4 Operating environment

2.5 User environment

2.6 Design/implementation constraints

2.7 Assumptions and dependencies

3. External Interface Requirements

3.1 User interfaces

3.2 Hardware interfaces

3.3 Software interfaces

3.4 Communication protocols and interfaces

4. System Features

4.1 System feature A

4.1.1 Description and priority

4.1.2 Action/result

4.1.3 Functional requirements

4.2 System feature B

5. Other Nonfunctional Requirements

5.1 Performance requirements

5.2 Safety requirements

5.3 Security requirements

5.4 Software quality attributes

5.5 Project documentation

5.6 User documentation

6. **Other Requirements Appendix A:** Terminology/Glossary/Definitions list **Appendix B:** To be determined

Conclusion: The Requirement Analysis and SRS was made successfully by following the steps described above

Draw E-R Diagram for Course Registration system.

This ER (Entity Relationship) Diagram represents the model of Course Registration System Entity. The entity-re lationship diagram of Course Registration System shows all the visual instrument of database tables and the rela tions between Fees, Students, Course, Trainers etc. It used structure data and to define the relationships between structured data groups of Course Registration System functionalities. The main entities of the Course Registra tion System are Course, Fees, Syllabus, Students, Registrations and Trainers.

Course Registration System entities and their attributes:

• Course Entity: Attributes of Course are course_id, course_student_id, course_registration, course_name, course_type, course_year, course_description • Fees Entity: Attributes of Fees are fees_id,

fees_amount, fees_type, fees_description

• Syllabus Entity: Attributes of Syllabus are syllabus_id, syllabus_course_id, syllabus_name, syllabus_type, syllabus_description

• Students Entity: Attributes of Students are student_id, student college_id, student_name, student_mobile, student email, student_username, student_password, student address

• Registrations Entity: Attributes of Registrations are registration_id, registration_student_id, registration_name, registration_type, registration number, registration_date, registration_description

• Trainers Entity: Attributes of Trainers are trainer_id, trainer_course_id, trainer_name, trainer_mobile, trainer_email, trainer_ username trainer_password, trainer_address

Description of Course Registration System Database:

• The details of Course is store into the Course tables respective with all tables

• Each entity (Trainers, Syllabus, Registrations, Fees, Course) contains primary key and unique keys.

• The entity Syllabus, Registrations has binded with

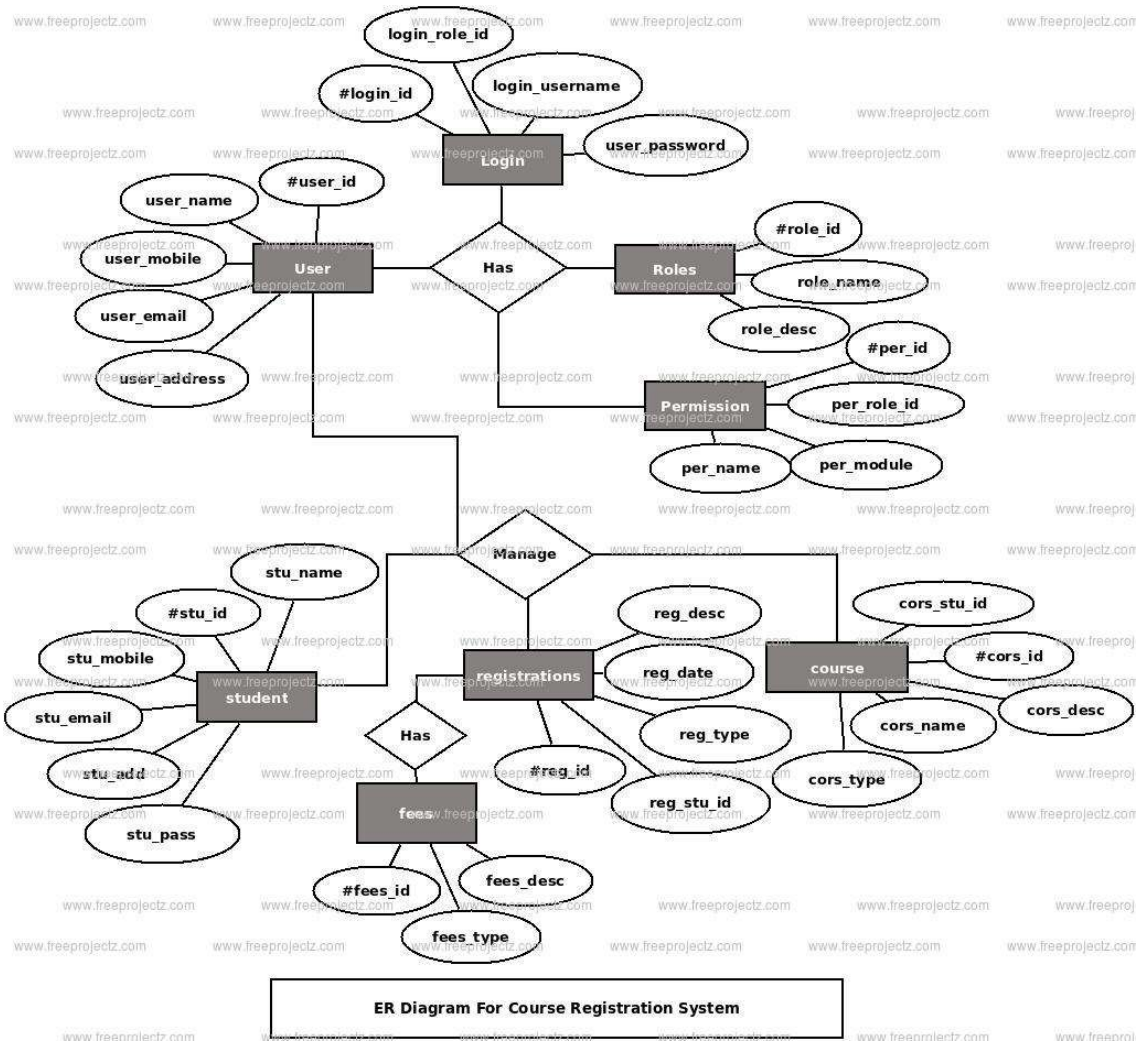
Course, Fees entities with foreign key

• There is one-to-one and one-to-many relationships available between Registrations, Students, Trainers, Course

• All the entities Course, Registrations, Syllabus, Trainers

are normalized and reduce duplicacy of records

• We have implemented indexing on each tables of Course Registration System tables for fast query Execution.



ER Diagram For Course Registration System

Draw DFD for Course Registration system.

Course Registration System Data flow diagram is often used as a preliminary step to create an overview of the Course without going into great detail, which can later be elaborated.it normally

consists of overall application data flow and processes of the Course process. It contains all of the userflow and their entities such all the flow of Course, Fees, Syllabus, Students, Trainers, Registration, Login. All of the below diagrams has been used for the visualization of data processing and structured design of the Course process and working flow.

Zero Level Data Flow Diagram(0 Level DFD) Of Course Registration System :

This is the Zero Level DFD of Course Registration System, where we have eloborated the high level process of Course. It's a basic overview of the whole Course Registration System or process being analyzed or modeled. It's designed to be an at-a-glance view of Trainers,Registration and Login showing the system as a single high-level process, with its relationship to external entities of Course,Fees and Syllabus. It should be easily understood by a wide audience, including Course,Syllabus and Trainers In zero leve DFD of Course Registration System, we have described the high level flow of the Course system.

High Level Entities and proccess flow of Course Registration System:

• Managing all the Course

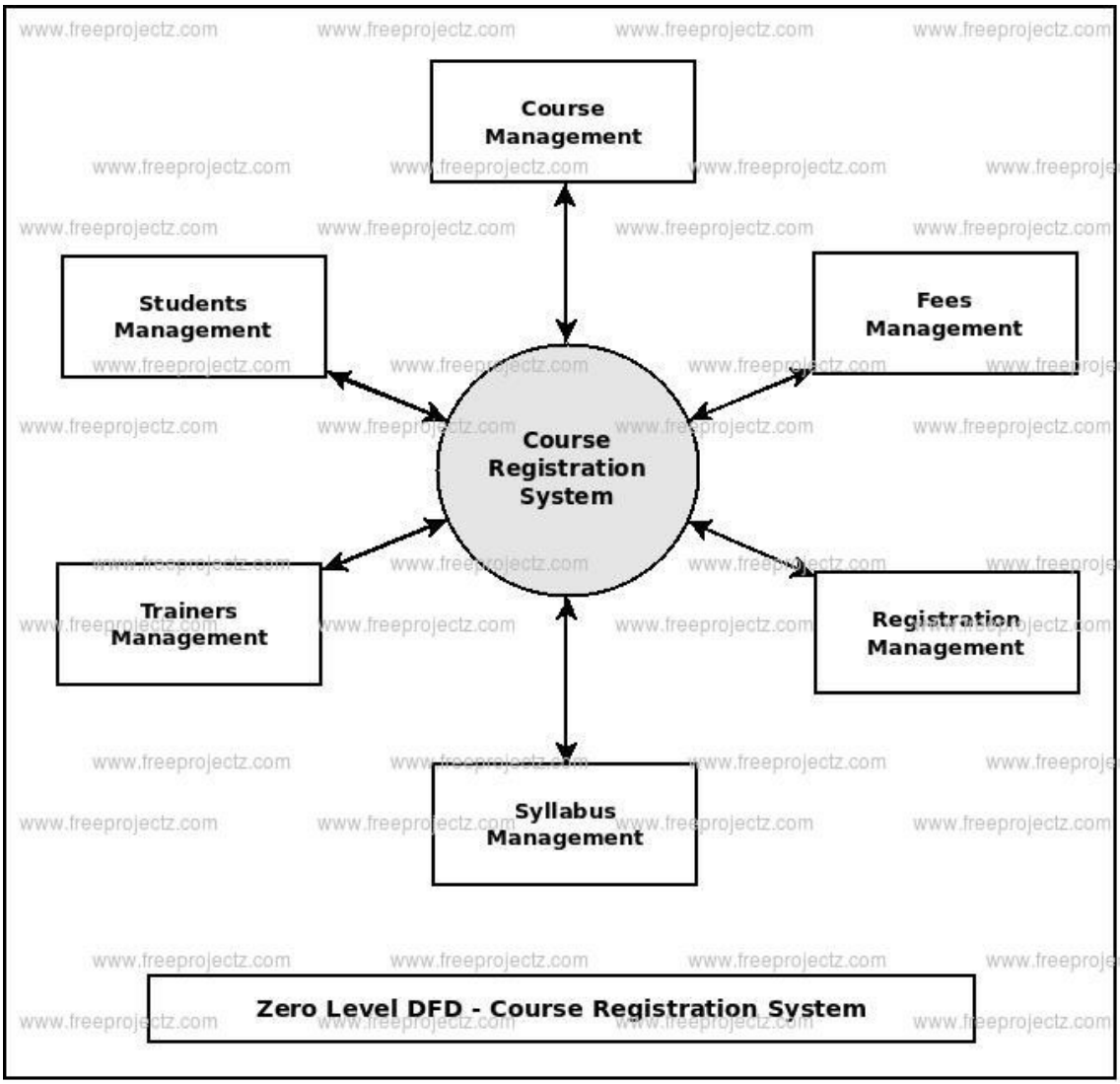
• Managing all the Fees

• Managing all the Syllabus

• Managing all the Students • Managing all the Trainers

• Managing all the Registration

• Managing all the Login



First Level Data Flow Diagram(1st Level DFD) Of Course Registration System :

First Level DFD (1st Level) of Course Registration System shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the Course Registration System system as a whole. It also identifies internal data stores of Login, Registration, Trainers, Students, Syllabus that must be present in order for the Course system to do its job, and shows the flow of data between the various parts of Course, Syllabus, Registration, Login, Trainers of the system. DFD Level 1 provides a more detailed breakout of pieces of the 1st level DFD. You will highlight the main

functionalities of Course.

Main entities and output of First Level DFD (1st Level DFD):

• Processing Course records and generate report of all

Course • Processing Fees records and generate report of all

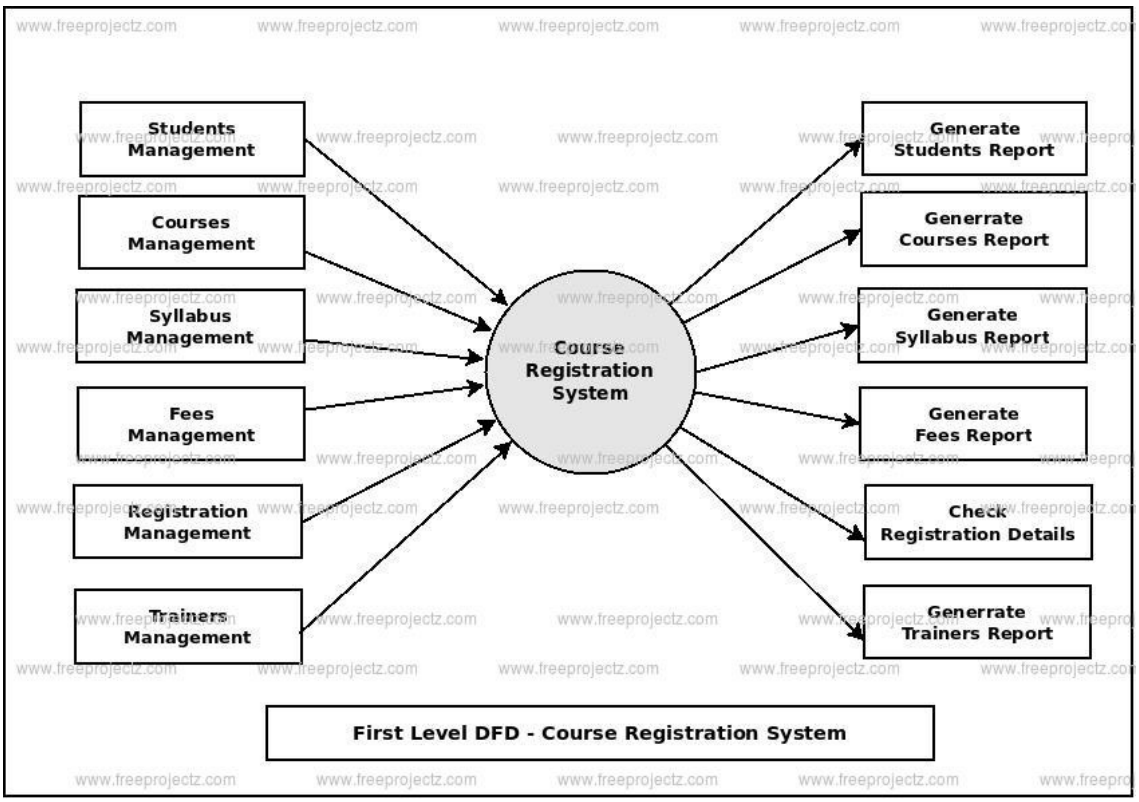
Fees • Processing Syllabus records and generate report of all

Syllabus

Students • Processing Trainers records and generate report of all

• Processing Students records and generate report of all Trainers • Processing Login records and generate report of all Login

• Processing Registration records and generate report of all Registration



Second Level Data Flow Diagram(2nd Level DFD) Of Course Registration System:

DFD Level 2 then goes one step deeper into parts of Level 1 of Course. It may require more functionalities of Course to reach the necessary level of detail about the Course functioning. First Level DFD (1st Level) of Course Registration System shows how the system is divided into subsystems (processes). The 2nd Level DFD contains more details of Login, Registration, Trainers, Students, Syllabus, Fees, Course.

Low level functionalities of Course Registration System

• Admin logins to the system and manage all the

functionalities of Course Registration System • Admin can add, edit, delete and view the records of Course, Syllabus, Trainers, Login

• Admin can manage all the details of Fees, Students,

Registration • Admin can also generate reports of Course, Fees, Syllabus, Students, Trainers, Registration • Admin can search the details of Fees,

Trainers, Registration

• Admin can apply different level of filters on report of Course, Students, Trainers

• Admin can tracks the detailed information of Fees, Syllabus, Students,, Trainers