## \Project flow:

What is project?

Project is the development of software system, to automate a set of activities or tasks.

Project

Activities

Resources

Work products

Tasks

Time, equipment, participants, roles

System, model and documents

Abstract:

## Purpose

Implement a system that handles all the tasks of the integrated services and the companies that are registered to manage the employee recruitment with instant verification of his/her previous working experience.

Objective of our project:

System Analysis:

-Existing System.

-Proposed System.

Module Description:

1. Administrator Interface Design.
2. User Interface.
3. Security Authentication.
4. Reports.
5. General end-users.

**SOFTWARE REQUIREMENTS**

Operating System : Windows XP/or 2003 Linux

User Interface : HTML, CSS

Client-side Scripting : JavaScript

Programming Language : Java

Web Applications : JDBC, Servlets, JSP

IDE/Workbench : My Eclipse 8.6

Database : Oracle 10g/11g

Server Deployment : Tomcat 6.x/7.x

**HARDWARE REQUIREMENTS**

Processor : core2Duo

Hard Disk : 160GB

RAM : 1GB or more

HTML, CSS (cascading style sheets) **:** for developing presentation logic(GUI).

My Eclipse (IDE-Integrated Development Environment) **:** It makes easy to develop. If there are any errors we can rectify them easily.

Oracle 10g **:** There are 12 E F Codd’s rules in that 11 rules satisfies by Oracle.

And 6 rules satisfy by RDBMS (MYSQL).

JavaScript **:** Validations and for Calculation purposes.

JavaScript is most commonly used as a client side scripting language. This means that JavaScript code is written into an HTML page. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it's up to the browser to do something with it.

Tomcat **:** Local server to develop.

SDLC

The software/system development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process. SDLC is a structure followed by a development team within the software organization. It consists of a detailed plan describing how to develop, maintain and replace specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

Sdlc consists of 5 phases

1. Requirement and analysis
2. Design
3. coding
4. Testing
5. Implementation and maintenance

Each stage produces a set of deliverables needed in the next stage. Requirements are analyzed, and the functional requirements are determined, based on the functional requirements needed, the system is designed and code is produced during implementation phase. Testing phase verifies the deliverables from the implementation phase meet the functional requirements or not.

Need for sdlc :

1. Using sdlc, the roles are clearly assigned to the participants of the project which saves the time and increases the efficiency of the system.
2. The system can be maintained well and modularity is achieved.
3. For complex projects, sdlc helps to reach the goals in specified time.

**Analysis:**

During analysis phase, the participants of the system, i.e. clients, project managers and business analysts gather together in business meeting. Client gives the data and specifies his requirements. The business analyst will analyze the data and prepares a BRS document (business requirement specification).

From the BRS document FRS is produced, which describes the functional and non – functional requirements of the system. It determines how the system should perform, what are the inputs and outputs of the system, who are the users of the system.

The project manager will take over the FRS and prepares a SRS document (system Requirement specification). At this stage, project manager assigns the tasks to team leads and their coordinates.

The test Lead will prepare a test plan to test the system. The BRS, FRS, SRS and test plan are again verified by the Quality analyst. He compares the client specifications and the BRS and fills any gaps/ loop holes in the analysis part. The business analyst finally prepares the use case document from which the system is designed.

Requirement analysis …………………………………

Feasibility Report:

-Technical Feasibility

- Operational Feasibility

- Economical Feasibility

**Feasibility report**: it estimates the ability to complete the project . it is categorized into 3 types

1. Technical feasibility :

it analyses what are the technologies are needed for the development of proposed system . And whether the organization posses the required technology needs if yes, is the current capacity is sufficient or not.

2. Operational feasibility:

It analyzes whether the system works as per the organizational requirements after it is developed and installed and also are there any difficulties to develop the system , is there good support from the users of the system , i.e. are they efficiently involved in the system .

3. Economic feasibility:

This study determines the cost to develop the proposed system and the benefits of replacing the existing system with the new system. The benefits include faster retrieval of the data, better customer satisfaction, efficient system on account of system automation etc.

**Sdlc methodologies** …. Examples water fall model, prototype and spiral model.

**Water fall model** … the first introduced model of sdlc. In this model, each phase must be completed fully before the next phase begins. Hence it is also called linear sequential model. If any problem occurs in the intermediate phases, again the process should begin from the staring. This is the main drawback of this model. If we want to make change in the project at testing stage it becomes very complex to do it.

[](http://istqbexamcertification.com/wp-content/uploads/2012/01/Waterfall-model.jpg)

**Disadvantages of waterfall model:**

* Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* No working software is produced until late during the life cycle.
* For high amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.

**When to use the waterfall model:**

* Requirements are very well known, clear and fixed without ambiguities.
* Product definition is stable.
* There are no ambiguous requirements
* Ample resources with required expertise are available freely
* The project is short.

**Prototype model**:

The basic idea here is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an “actual feel” of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.  Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements. The prototypes are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

**Diagram of Prototype model:**

[](http://istqbexamcertification.com/wp-content/uploads/2012/01/Prototype-model.jpg)

**Advantages of Prototype model:**

* Users are actively involved in the development
* Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
* Errors can be detected much earlier.
* Quicker user feedback is available leading to better solutions.
* Missing functionality can be identified easily
* Confusing or difficult functions can be identified  
  Requirements validation, Quick implementation of, incomplete, but  
  functional, application.

**Disadvantages of Prototype model:**

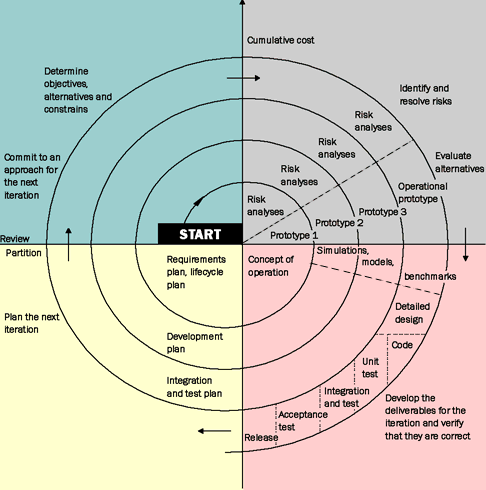
* Leads to implementing and then repairing way of building systems.
* Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
* Incomplete application may cause application not to be used as the  
  full system was designed  
  Incomplete or inadequate problem analysis.
* **When to use Prototype model:**
* Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
* Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
* Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

**Spiral Model:**

The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. **Requirements** are gathered during the planning phase.  In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions.  A prototype is produced at the end of the risk analysis phase.

Software is produced in the **engineering phase**, along with testing at the end of the phase.  The **evaluation phase** allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

**Diagram of Spiral model:**



**Advantages of Spiral model:**

* High amount of risk analysis hence, avoidance of Risk is enhanced.
* Good for large and mission-critical projects.
* Strong approval and documentation control.
* Additional Functionality can be added at a later date.
* Software is produced early in the software life cycle.

**Disadvantages of Spiral model:**

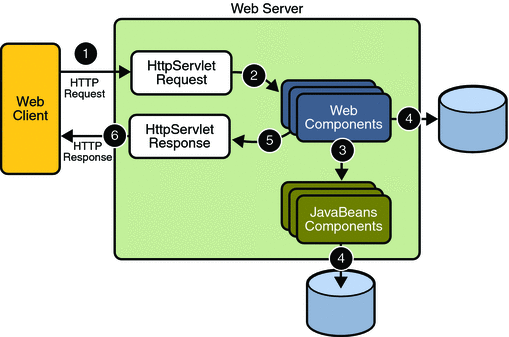
* Can be a costly model to use.
* Risk analysis requires highly specific expertise.
* Project’s success is highly dependent on the risk analysis phase.
* Doesn’t work well for smaller projects.

**When to use Spiral model:**

* When costs and risk evaluation is important
* For medium to high-risk projects
* Long-term project commitment unwise because of potential changes to economic priorities
* Requirements are complex
* New product line.
* Significant changes are expected (research and exploration).

MVC ARCHITECTURE

USER VIEW CONTROLLER MODEL



The standard architecture for project design is MVC architecture.

The standard architecture for project development is LC-RP(loosly coupled runtime polymorphysm) architecture.

MVC Definition:

It stands for Model View Controller.MVC is a standard architectural design pattern for developing World Wide Web apps.The central idea behind MVC is code reusability and separation of roles of code development.

In every project development we find three types of applications.

An application for

1. Performing calculations and CRUD operations.

2. Displaying GUI and Report.

3. Integrating above applications.

So every project must be divided into three layers i.e., Model View Controller.

Explanation:

1. First end-user downloads GUI from server (i.e., .html file).

2. Next, sends input to controller application (server) through that GUI.

3. Then controller based on the request type calls Model (jdbc application).

4. Then Model performs CRUD operations on DB, generates output, sends that output back to Controller.

5Then Controller sends that output to report generation view application (jsp).

6. Then Report view application (jsp) updates the Model gives output in the html template and sends that updated template back to Controller.

7. Finally Controller sends that dynamically prepared html browser.

8. Browser displays that response to end-user.

Why it is named as MVC?

First DB design is followed by Model components, next View components to collect input and to display response and finally Controller components to integrate View and Model components.

Advantages:

Using this architecture we can clearly separate roles so that parallel development is possible. Hence we can complete project in short time. We can easily enhance project for adding new features.

1. **THE PRESENTATION LAYER**

Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

1. **THE BUSINESS RULES LAYER**

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process

1. **THE DATA ACCESS LAYER**

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

1. **THE DATABASE LAYER**

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc.   
In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

**Design:**

At this level, the architecture of the system is determined. The components of design level are

1. Software and hardware requirements
2. How Communication should be done between hardware and software
3. DFD’s and UML diagrams.

**system design . . . .**

In designing part first we consider the data flow diagrams ….

Data flow diagram: it is the graphical representation of the data flowing through the system, what are the inputs of the system and outputs of the system and how the data is stored in the system.

Dfd is the first step carried out in developing any information system. The components of dfd are

1 . data flow : the movement of data from source to destination is represented by data flow .

2 . process : the people or activities or procedures which transform inputs to outputs .

3. Data source: External sources or destination of data, which may be People programs, organizations or other entities.

4 . data stores : the data is stored in stores , and referenced by a process in the system .

Example:



dfd is categorized into

context level dfd : it is drawn based on the system requirements , i.e it describes the basic view of the system , what are the inputs , processes , how the data is flown through the system and to which the data store a process refers to.

0 level dfd: 0 level dfd are drawn by expanding the context level dfd’s. This dfd contains the details of the subsystems and the data flow in the sub systems.

First level:

Second level:

Example:

**DATA FLOW DIAGRAM**

**Login**

1.0

Login

1.1

Administrator

1.2

Client

User id, Password &dept type

User id & Password

**Administrator**

1.1.5

Certficate Admin

1.1.4

Tax Admin

1.1.3

MCH Admin

1.1.2

Telephone Admin

1.1.1

User,position,activity

1.1

Administrator

Select Type

1.1.1

EAdmin

1.1.1.5

View Payment Report

1.1.1.4

Connect Service

1.1.1.3

Disconnect Service

1.1.1.2

Deposit Amount

1.1.1.1

Bill Generation

Customers

Deposits

Bills

Get cust info

Update

Retrieve

Update

Update

Update

ReportPayments

UML diagrams ….

What is uml ?

Uml means unified modeling language which is used for specifying, visualizing, constructing and documenting the requirements of software system. Uml is mainly used in object oriented software design.

Uml mainly determines the structural and behavioral view of the system.

uml

structural

behavioural

activity

Use case

sequence

Class diagrams

Deployement

diagrams

Componenet collaboration diagram

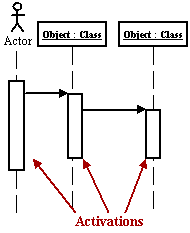
diagram

Sequence diagram :

These diagrams explains the message exchanged between the various objects of the system .

For example ….. if a agent login with uname and password , loginAction() method is called . if it executes successfully , next method is called or else , the loginAction() is called again .

Class roles



Life line

messages

Class roles : class roles describe the behavior of the object within the system .

Activations : the time taken by the object to complete a task

Life line : the object presence over the time is represented by lifeline .

**Collaboration Diagram**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

**What is a UML Class Diagram?**

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

Classes are used to represent objects. Objects can be anything having properties and responsibility.

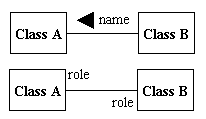
Here the symbols + , -, # are the visibility specifiers . when an attribute is declared as public , it can be accessed from within the package and outside the package .

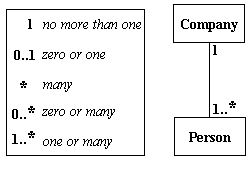
When declared as private , it can be accessed within the class only

When declared as protected , the attribute can be accessed within the package and child class of another class..

**Associations**

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.  
***Note:*** It's uncommon to name both the association and the class roles.

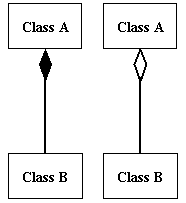




http://wc1.smartdraw.com/resources/tutorials/images/uml_constraint.gif*Simple Constraint*

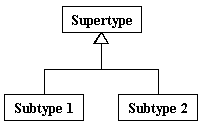
**Composition and Aggregation**

Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate **composition** with a filled diamond. Use a hollow diamond to represent a simple **aggregation** relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



**Generalization**

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.



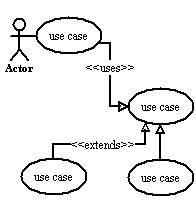
In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

Use case diagrams:

They describe the functionality of the system from user’s point of view. They focus on the behavior of the system from external point of view. Here the actors interact with the system use cases to perform specific tasks. Actors are the users of the system and use cases are the tasks that they perform on the system.

**Relationships**

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.





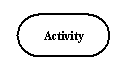
**Activity Diagram**

An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses some of the same modeling conventions.

**Basic Activity Diagram Symbols and Notations**

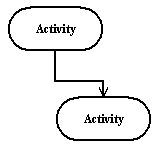
**Action states**

Action states represent the non interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.



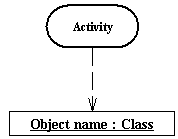
**Action Flow**

Action flow arrows illustrate the relationships among action states.



**Object Flow**

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



**Initial State**

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

Initial State

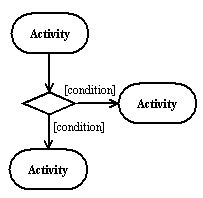
**Final State**

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

Final State

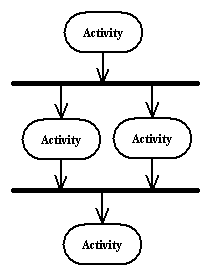
**Branching**

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



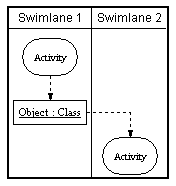
**Synchronization**

A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



**Swimlanes**

Swimlanes group related activities into one column.



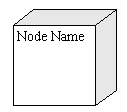
**What is a UML Deployment Diagram?**

Deployment diagrams depict the physical resources in a system including nodes, components, and connections.

**Basic Deployment Diagram Symbols and Notations**

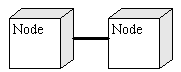
**Component**

A node is a physical resource that executes code components.  
[Learn how to resize grouped objects like nodes.](http://www.smartdraw.com/resources/tutorials/Objects)



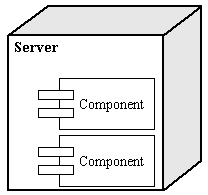
**Association**

Association refers to a physical connection between nodes, such as Ethernet.  
[Learn how to connect two nodes.](http://www.smartdraw.com/resources/tutorials/Lines)



**Components and Nodes**

Place components inside the node that deploys them.



**What is normalization?**

**Definition:** Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored. There are several benefits for using Normalization in Database.

**Benefits:**

1. Eliminate data redundancy
2. Improve performance
3. Query optimization
4. Faster update due to less number of columns in one table
5. Index improvement

There is diff. - diff. types of Normalizations form available in the Database. Let’s see one by one.

1. First Normal Form (1NF)

 First normal form (1NF) sets the very basic rules for an organized database:

* Eliminate duplicative [columns](http://databases.about.com/library/glossary/bldef-column.htm) from the same table.
* Create separate tables for each group of related data and identify each [row](http://databases.about.com/library/glossary/bldef-row.htm) with a unique column or set of columns (the [primary key](http://databases.about.com/library/glossary/bldef-primarykey.htm)).
  1. Remove repetitive groups
  2. Create Primary Key

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **Name** | **State** | **Country** | **Phone1** | **Phone2** | **Phone3** |
| John | 101 | 1 | 488-511-3258 | 781-896-9897 | 425-983-9812 |
| Bob | 102 | 1 | 861-856-6987 |  |  |
| Rob | 201 | 2 | 587-963-8425 | 425-698-9684 |  |
| **PK** |  |  | **[ Phone Nos ]** | | |
| ? |  |  |  | ? |  |
| **ID** | **Name** | **State** | **Country** | **Phone** |  |
| 1 | John | 101 | 1 | 488-511-3258 |  |
| 2 | John | 101 | 1 | 781-896-9897 |  |
| 3 | John | 101 | 1 | 425-983-9812 |  |
| 4 | Bob | 102 | 1 | 861-856-6987 |  |
| 5 | Rob | 201 | 2 | 587-963-8425 |  |
| 6 | Rob | 201 | 2 | 425-698-9684 |  |
|  |  |  |  |  |  |

2. **Second Normal Form (2NF)** second normal form (2NF) further addresses the concept of removing duplicative data:

·         Meet all the requirements of the first normal form.

·         Remove subsets of data that apply to multiple rows of a table and place them in separate tables.

·         Create relationships between these new tables and their predecessors through the use of [foreign keys](http://databases.about.com/library/glossary/bldef-foreignkey.htm).

Remove columns which create duplicate data in a table and related a new table with Primary Key – Foreign Key relationship

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Name | State | Country | Phone |  |  |  |
| 1 | John | 101 | 1 | 488-511-3258 |  |  |  |
| 2 | John | 101 | 1 | 781-896-9897 |  |  |  |
| 3 | John | 101 | 1 | 425-983-9812 |  |  |  |
| 4 | Bob | 102 | 1 | 861-856-6987 |  |  |  |
| 5 | Rob | 201 | 2 | 587-963-8425 |  |  |  |
| 6 | Rob | 201 | 2 | 425-698-9684 |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| ID | Name | State | Country |  | PhoneID | ID | Phone |
| 1 | John | 101 |  |  | 1 | 1 | 488-511-3258 |
| 2 | Bob | 102 |  | 2 | 1 | 781-896-9897 |
| 3 | Rob | 201 |  | 3 | 1 | 425-983-9812 |
|  |  |  |  |  | 4 | 2 | 587-963-8425 |
|  |  |  |  |  | 5 | 3 | 587-963-8425 |
|  |  |  |  |  | 6 | 3 | 425-698-9684 |

3. **Third Normal Form (3NF)**

Third normal form (3NF) goes one large step further:

·         Meet all the requirements of the second normal form.

·         Remove columns that are not dependent upon the primary key.

  Country can be derived from State also… so removing country

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | State | Country |
| 1 | John | 101 | 1 |
| 2 | Bob | 102 | 1 |
| 3 | Rob | 201 | 2 |

**4. Fourth Normal Form (4NF)**

Finally, fourth normal form (4NF) has one additional requirement:

·         Meet all the requirements of the third normal form.

·         A relation is in 4NF if it has no multi-valued dependencies.

If PK is composed of multiple columns then all non-key attributes should be derived from FULL PK only. If some non-key attribute can be derived from partial PK then remove it

The 4NF also known as BCNF NF

|  |  |  |  |
| --- | --- | --- | --- |
| TeacherID | StudentID | SubjectID | StudentName |
| 101 | 1001 | 1 | John |
| 101 | 1002 | 2 | Rob |
| 201 | 1002 | 3 | Bob |
| 201 | 1001 | 2 | Rob |
|  |  |  |  |
|  |  |  |  |
| TeacherID | StudentID | SubjectID | StudentName |
| 101 | 1001 | 1 | X |
| 101 | 1002 | 2 | X |
| 201 | 1001 | 3 | X |
| 201 | 1002 | 2 | X |

Testing:

During testing, the implementation is tested against the Requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. Unit tests and system/acceptance tests are done during this phase. Unit tests act on a specific component of the system, while system tests act on the system as a whole.

Implementation:

Code is produced from the design in this phase. Many tools exist to automate the production of code from the design. These are called computer aided software engineering (CASE) tools