

Design Concepts and Design Principles

- સોફ્ટવેર ડિઝાઇન ને સોફ્ટવેર એન્જિનિયરિંગ નું technical core કહેવાય છે અને ઉપયોગમાં લેવાતા સોફ્ટવેર પ્રોસેસ મોડલને ધ્યાનમાં લીધા વગર લાગુ કરવામાં આવે છે.
- ડિઝાઇન task ડેટા ડિઝાઇન, આર્કિટેક્ચરલ ડિઝાઇન, ઇન્ટરફેસ ડિઝાઇન અને કોમ્પોનેન્ટ ડિઝાઇન બનાવે છે.

Database Design

- જ્યાં તમે System ની માહિતી ની Design કેવી રીતે કરવામાં આવે છે તે Database માં રજૂ કરી છે.
- તેમાં Data Object અને Relationship E-R Diagram પરથી Define કરવામાં આવે છે.
- વિગતવાર માહિતી Data Dictionary માં દર્શાવવામાં આવે છે.

Architectural design :

- જ્યાં તમે System નું Overall Structure ઓળખવા ના મુખ્ય ઘટકો (ક્યારેક Sub-System અથવા Modules) કહેવાય છે.
- અને તેને કેવી રીતે વહેંચવામાં આવે છે તે DFD પરથી Derived કરવામાં આવે છે.

Interface design :

- જ્યાં તમે System Components વચ્ચે Interface કરે છે.
- તે System પોતે અને Users સાથે વાતચીત કેવી રીતે કરશે તે વર્ણવે છે.
- તે DFD અને State Transition Diagram પરથી Derived કરવામાં આવે છે.

Component-level design

- Design analysis model traceable હોવી જોઈએ.
- Design ને સોફ્ટવેર અને problem વચ્ચેનું “intellectual distance ઓછું કરવું” જોઈએ કારણ કે તે real world માં exists હોઈ છે.
- Design માં uniformity અને integration હોવું જોઈએ.
- Design change માટે structured માં હોવી જોઈએ.
- જ્યારે abnormal data, events અથવા ઓપરેટિંગ conditions સામનો કરવો પડે ત્યારે પણ ડિઝાઇનને gently degrade કરવા માટે structured કરવી જોઈએ.
- ડિઝાઇન એ કોડિંગ નથી, અને કોડિંગ એ ડિઝાઇન નથી.
- ડિઝાઇન નું assessed quality માટે થવી
- Conceptual (semantic) ભૂલોને ઘટાડવા માટે ડિઝાઇન reviewed થવી જોઈએ.

Component Level Design (Function Oriented Design, Object Oriented Design)

Methodology ની પ્રકૃતિ, Users, Qualification અને Training Software Develop Team સાથે, ઉપલબ્ધ હાર્ડવેર અને સોફ્ટવેર સાધનોની સંખ્યા પર આધાર રાખીને વિકસાવવામાં આવ્યું છે.

- તેના Two Different Types છે :
 1. Function oriented design: it can be further divided in two category
 - I. Structured Analysis

II. Structured Design

2. Object oriented design

Function oriented design (Top-Down approach)

- Functional Oriented Design માં Set of Function Perform કરવામાં આવે છે.
- System ના High Level ના શરૂઆતથી, બધા Function define એક કરતાં વધુ Function માં Detail માં થાય છે. તે દરેક Sub function માં વિભાજિત થાય છે.
- તે બે Categories માં Divide કરવામાં આવે છે.

Structured Analysis

- તેનો ઉપયોગ Textual Problem ને Graphical Form માં પરીવર્તન કરે છે.
- તે System નું વિગતવાર માળખું ચકાશે છે.
- તે Process વચ્ચે Process અને Data નો Flow દર્શાવે છે.
- Structure Analysis માં Functional Requirements SRS માં DFD થી Data નો Flow બતાવે છે.

Structured Design

- Structure Design દરમિયાન તમામ Function Analysis Time એ ઓળખીને Module બનાવવામાં આવે છે અને તે Implementation માટે ઉપયોગી છે.
- Structure Design નો મુખ્ય ઉદ્દેશ Structure Analysis ના પરિણામને Structure Chart માં પરીવર્તન કરે છે.

Object oriented design

- Object Oriented Design ની અંદર System Object ના સમૂહ થી જોવામાં આવે છે.
- Object Oriented Design Abstraction, hiding અને modularity જેવા Concept ને Support કરે છે.
- Design એ Program Domain તરફથી Solution Domain તરફ આગળ વધી નું Initial Step છે.
- Design નો મુખ્ય હેતુ Working Solution એવું કરવાનું કે જે સરળતાથી Programming Language Code માં Translate કરી શકાય.
- UML Modeling અને Use Case Diagram નો ઉપયોગ Object Oriented Designing માં થાય છે.

E-R diagram

- E-R Diagram એ Real-World Entities ના Set અને તેમની વચ્ચેના Relationship ને દર્શાવે છે.
- એકવાર ER Diagram બનાવવામાં આવે તેની માહિતી Database ની અંદર Store કરવામાં આવે છે.
- E-R Diagram ની અંદર Data Object અને Entities, Data Attributes, Relationship, અને Cardinality નો સમાવેશ થાય છે.

Data object

- Data Object એ Real Word Entity છે.
- Data Object એ સંયુક્ત માહિતી જે Software માં ઉપયોગમાં લેવામાં આવે છે તેને રજૂ કરે છે.

- સંયુક્ત માહિતી વિવિધ સુવિધાઓ અથવા Data Object ના Attribute નો ઉલ્લેખ કરે છે.
- Entity એ System વિશેની માહિતી ને Database માં Store કરે છે. Example, Student એક Entity છે.
- Entities Rectangle વડે દર્શાવાય છે, Attribute ને Ellipse વડે દર્શાવાય છે અને Relationship ને Diamond Symbol વડે દર્શાવાય છે.

Data attributes

- Data Attribute માં Data Object ની Properties ને Describe કરવામાં આવે છે.
- For Example, Account Entity ના Attribute 'Number', 'Balance', and So on...

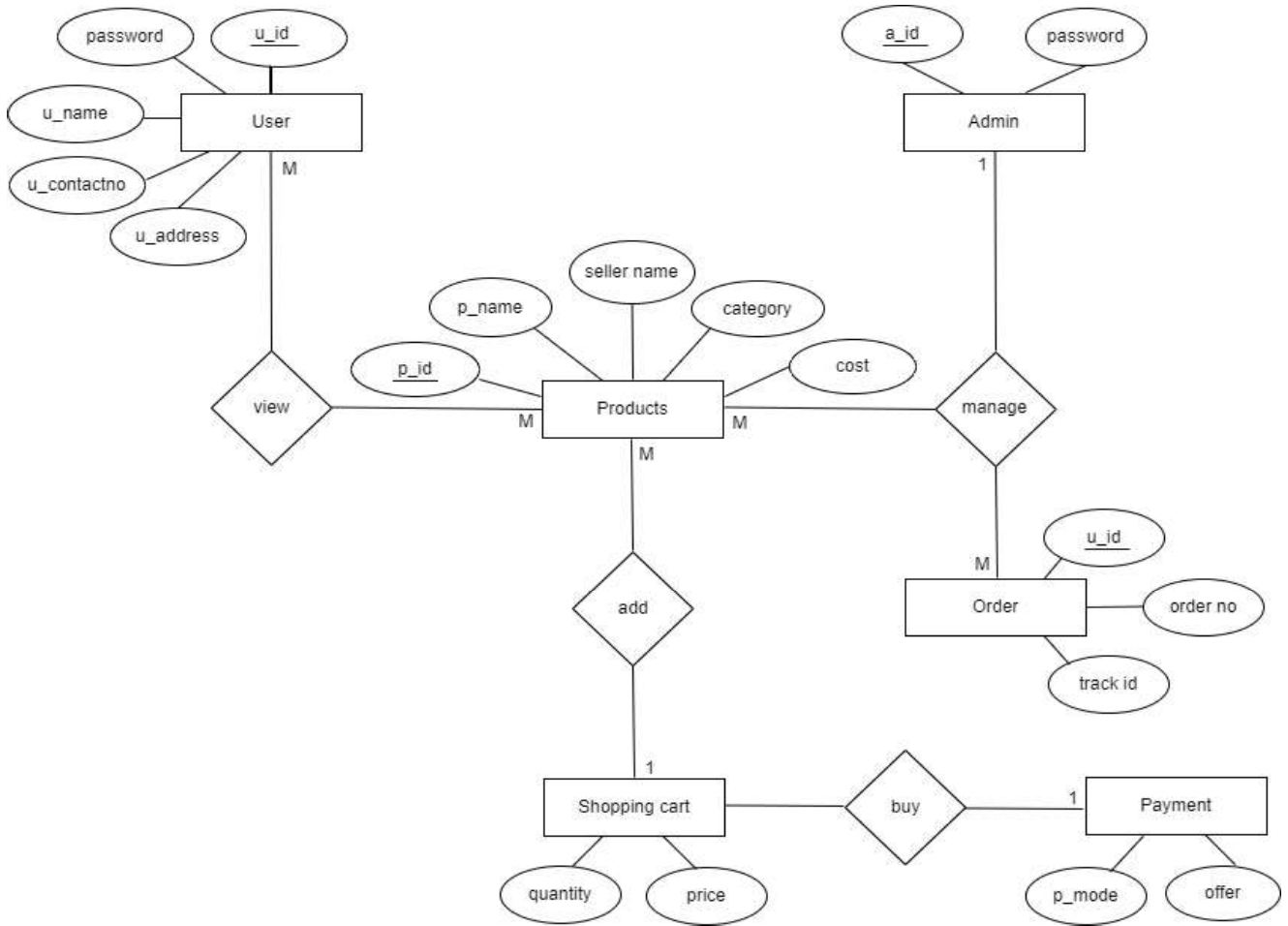
Relationships

- Entities અલગ-અલગ રીત થી એકબીજા સાથે Link કરવામાં આવે છે આ Link અને Data Object અને Entities ના Connection ને Relationship તરીકે ઓળખાય છે.
- Relationship દર્શાવવા માટે બે Entities હોવી જરૂરી છે.
- પહેલા Entities ઓળખાય પછી Development Team check કરે છે કે તેમાં કોઈ Relationship આવેલી છે કે નહીં.
- Relationship ને Diamond Symbol થી દર્શાવવામાં આવે છે અને તેમાં તે Relationship નું નામ આપવામાં આવે છે.

Cardinality

- Cardinality એ કેટલા Data Object અને Entity બીજા કેટલા Data Object અને Entity સાથે જોડાયેલા છે. તે દર્શાવે છે. તે કેટલી Entity Relationship માં છે. એ પણ ઉલ્લેખ કરે છે.
- Different cardinalities are explained below:
 - ✓ **One-to-one (1:1):** કોઈપણ એક Entity માત્રને માત્ર અન્ય એક Entity સાથે Connect હોય છે. Example, એક Bank માં User નું માત્ર ને માત્ર એક Bank Account હોય છે.
 - ✓ **One-to-many (1: M):** કોઈપણ એક Entity એ અન્ય બીજી ઘણી Entity ની સાથે Connect હોય છે. For Example, એક User ના અલગ-અલગ Bank માં અનેક Account હોય છે.
 - ✓ **Many-to-many (M: M):** ઘણી Entity અન્ય બીજી ઘણી Entity સાથે Connect હોય છે. Example, ઘણા બધા Users ના ઘણી બધી Bank માં Account હોય છે.

E-R diagram for online shopping:






Use case diagram

- Use case એ એક પદ્ધતિ છે જે System બનાવનાર ને તે System ની જરૂરિયાતોને ઓળખવી, સમજવી અને Mangle કરવા માટે ઉપયોગ થાય છે.
- Use case માં એવા કાર્યો દર્શાવવા માં આવે છે જેમાં User Software નો ઉપયોગ અમુક ચોક્કસ શરતો હેઠળ કરે છે.
- Use case એ Use case Diagram વડે દર્શાવવામાં આવે છે. તે Actor વચ્ચેના સંબંધો System ની અંદર આપે છે.
- દરેક Use case Order ને સમજવા માટે કેવી રીતે એક System અન્ય System સાથે Communication કરે છે તે અલગ કિસ્સામાં પૂરી પાડે છે. Use case માં Software Implementation Detail આપવામાં આવતી નથી.
- Use case Diagram ની અંદર System ની બહાર શું Exist છે તે Actor દર્શાવે છે અને System દ્વારા શું થાવું

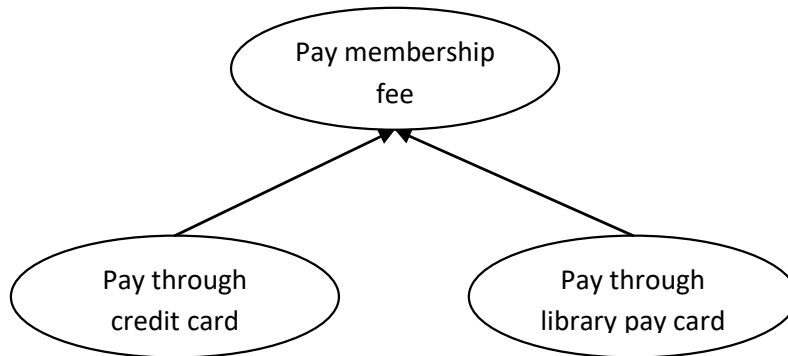
જોઈ એ Use case દર્શાવે છે.

Symbol Of Use Case :

Name	Description
Actor 	Actor એ અલગ અલગ પ્રકારના Users છે, જે System ને અલગ અલગ રીતે ઉપયોગ કરે છે .For Example, એક Actor Library User તથા બીજો User Library Staff હોય શકે.
Use case 	System નું કાર્ય વર્ણવે છે.
Communication link 	Actor અને User વચ્ચે પ્રતિક્રિયા સૂચવે છે.
User link	એક Use case નો ઉપયોગ બીજો Use case માં વર્ણવવામાં ઉપયોગી છે.

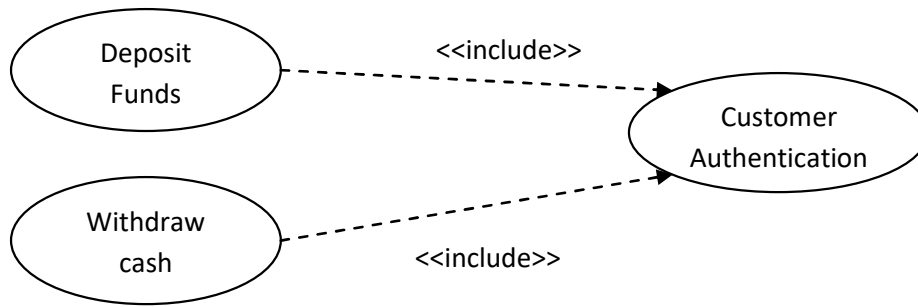
Generalization

- Use case ની અંદર Generalization નો ઉપયોગ ત્યારે કરવામાં આવે છે. જ્યારે એક Use case બીજાને સમાન હોય છે પરંતુ થોડું કઈક અલગ અને કઈક વધુ કરે છે.

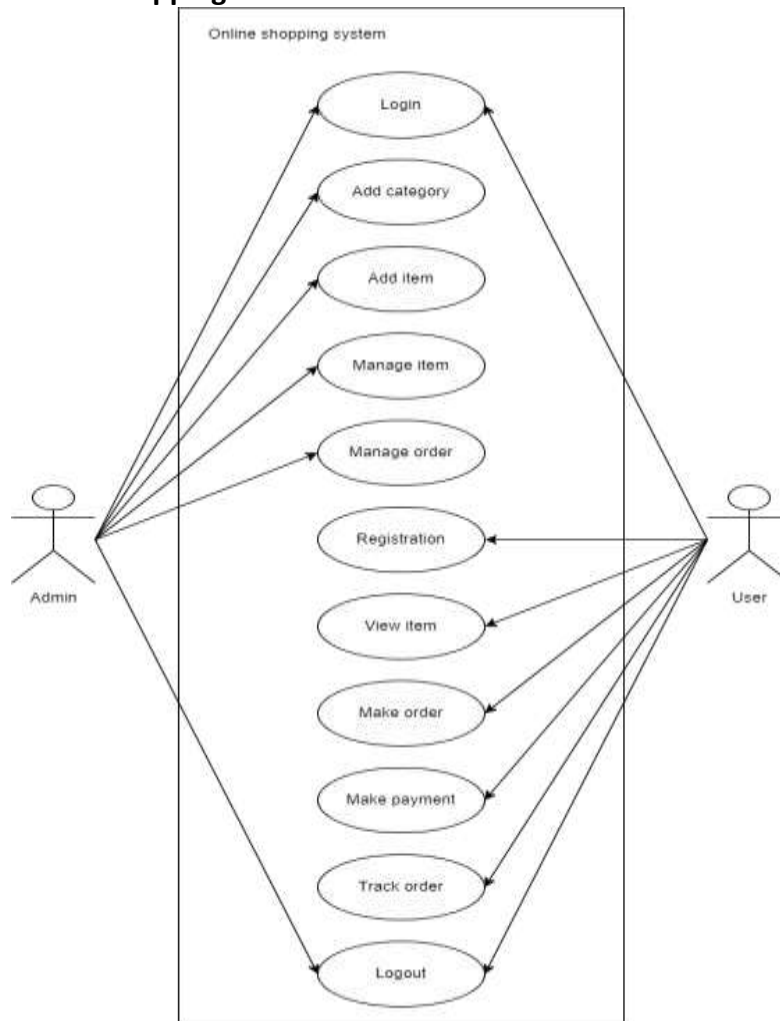


Includes

- Includes ની અંદર એક Use case ની અંદર અન્ય Use case નો સમાવેશ થાય છે. Include Relationship થાય છે જ્યારે તેના Behavior Use case ને સમાન હોય.
- Include ને <<include>> ની મદદ થી રજૂ કરવામાં આવે છે.



Use case diagram for online shopping



State diagram

- સ્ટેટ ડાયાગ્રામ એ એક ગ્રાફ છે જેના નોડ્સ સ્ટેટ્સ છે
- સ્ટેટ ડાયાગ્રામ state sequences ના ક્રમ ને આધારે event sequences ને specifies કરે છે.
- સ્ટેટ ડાયાગ્રામ એ standard કોમ્પ્યુટર સાયન્સ કોન્સેપ્ટ છે જે events અને state સંબંધિત છે.
- Class માંના તમામ ઓબ્જેક્ટ્સ તે class માટે સ્ટેટ ડાયાગ્રામ ચલાવે છે, જે તેમના common behavior મોડેલ બનાવે છે.
- સ્ટેટ ડાયાગ્રામ માટે નું નોટેશન એ upper left corner ના pentagonal ટેગમાં ઓબ્જેક્ટના નામ સાથેનો rectangle છે.

Components of state diagram

Initial state

- Outgoing એરો સાથેનું solid circle initial state દર્શાવે છે.



Final state

- Bull's eye – હોલો વર્તુળ/ઘેરાયેલ X દ્વારા બનેલું solid circle termination point દર્શાવે છે.



State

- State નું નામ ધરાવતા rounded box તરીકે દોરવામાં આવે છે.
- સ્ટેટ ડાયાગ્રામ માં State ના નામ unique હોવા જોઈએ.
- state નામને બોલ્ડફેસમાં કરવું, center માં name બાજુમાં top of the box માં રાખવું અને પ્રથમ અક્ષરને કેપિટલાઈઝ કરવું.

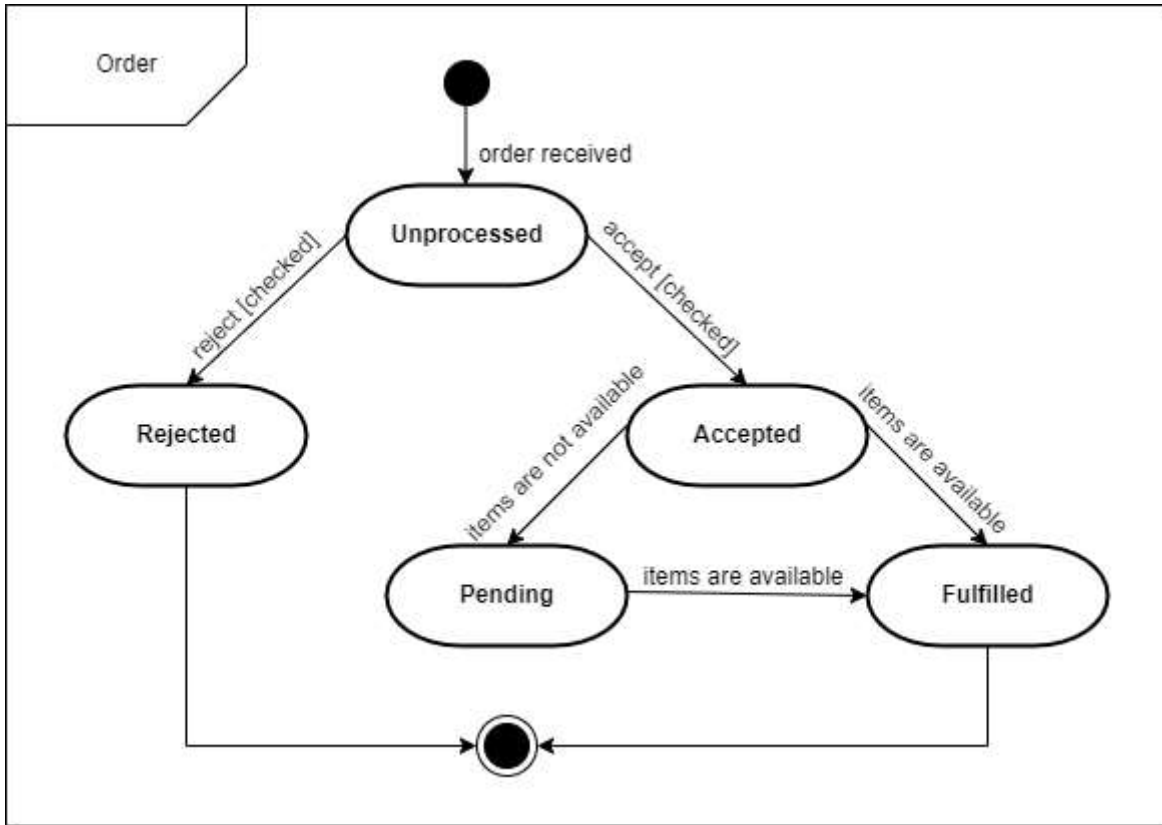


Transition / event

- Origin state થી target state સુધી line થી દોરવામાં આવે છે.
- એરોહેડ target state તરફ points કરે છે.



State diagram for online shopping



Activity Diagram

- Activity Diagram એ સામાન્ય રીતે એક Flow-Chart છે, જે એક Activity નો બીજી Activity સાથેનો Flow ને દર્શાવે છે.
- Activity Diagram System નો Dynamic Behavior દર્શાવે છે.
- Activity Diagram એક Activity થી બીજી Activity નો Message Flow બતાવે છે.

Basic Activity Diagram Symbols and Notations

Initial State

- ભરેલા Circle અને Arrow ને Initial State થી ઓળખાય છે.



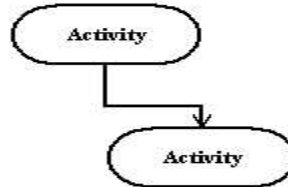
Action states

- Action States Object નું કાર્ય દર્શાવે છે.
- તેને Round Rectangle થી દર્શાવવામાં આવે છે.



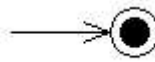
Action Flow

- Action Flow ની Arrow થી Activity ની સંબંધો દર્શાવે છે.



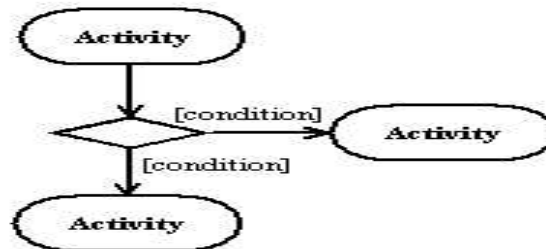
Final State

- એક Circle ની અંદર બીજું ભરેલું Circle માં એક Arrow Pointing કરે તેને Final State કરે છે.



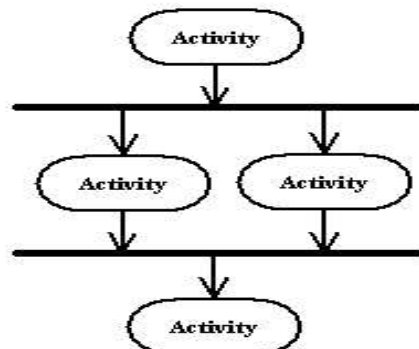
Branching

- Diamond વિકલ્પ સાથે નો નિર્ણય દર્શાવે છે. Outgoing માં એક Condition સાથે Label હોવું જોઈએ તમે એક ખોટા પાથ નું Label પણ રાખી શકો છો.

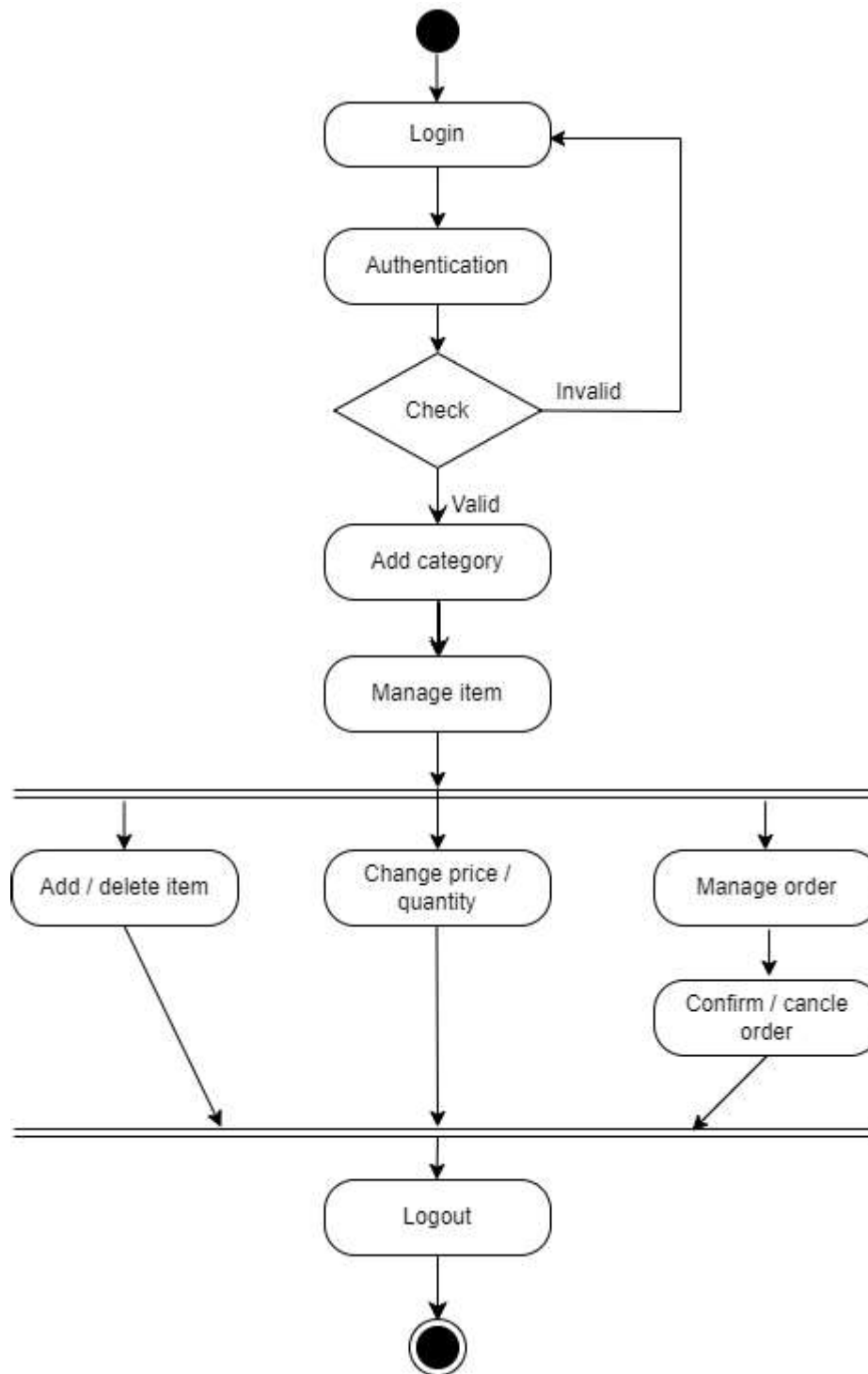


Synchronization

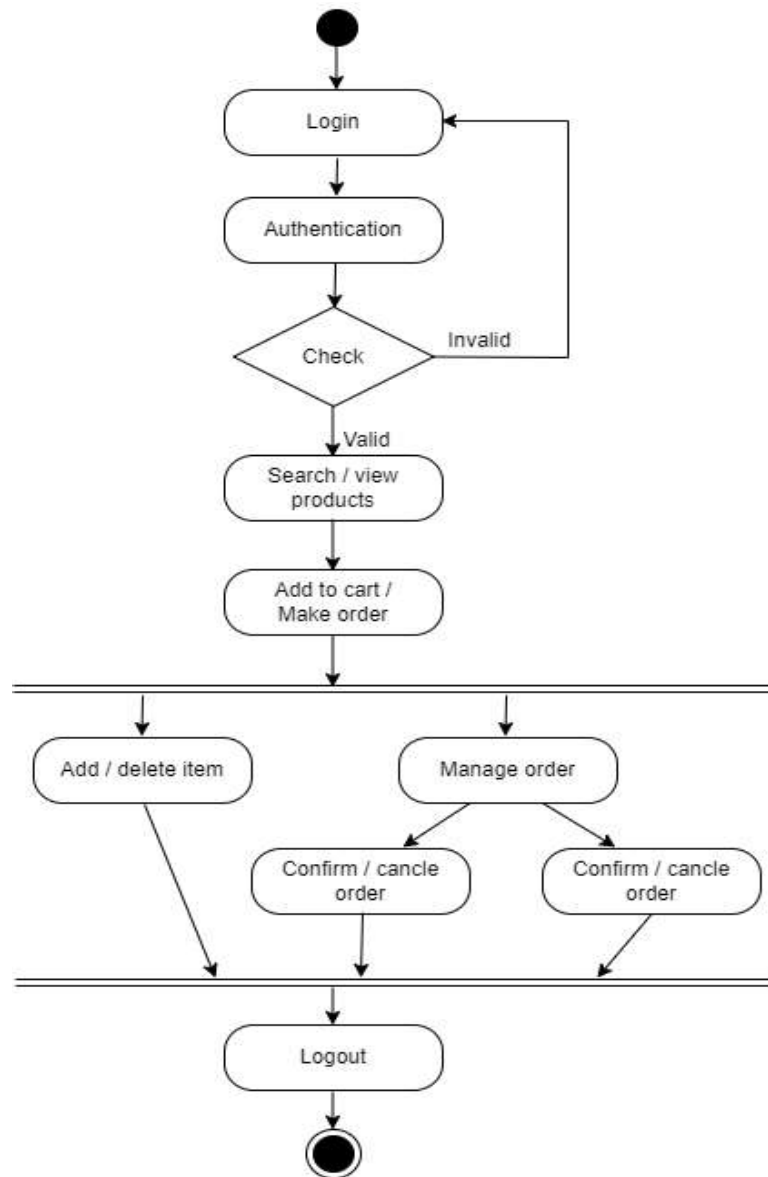
- સિંક્રોનાઇઝેશન બાર parallel transitions illustrate માં મદદ કરે છે. સિંક્રોનાઇઝેશનને ફોર્કિંગ અને જોઇનિંગ પણ કહેવામાં આવે છે.



Activity diagram for online shopping



Admin side Activity diagram



User side Activity diagram

Sequence Diagram

- સિક્વન્સ ડાયાગ્રામ માં ઓબ્જેક્ટ્સ ના group નો સમાવેશ થાય છે જે lifelines દ્વારા દર્શાવવામાં આવે છે અને messages કે જે તેઓ interaction દરમિયાન exchange કરે છે.
- સિક્વન્સ ડાયાગ્રામ નો ઉપયોગ મુખ્યત્વે ઓબ્જેક્ટ્સ વચ્ચે interactions જે તે sequential order થાય છે તે ક્રમમાં બતાવવા માટે થાય છે.
- સિક્વન્સ ડાયાગ્રામ કાર્યના execution દરમિયાન ઓબ્જેક્ટ વચ્ચે dynamic communication represent કરે છે.

- તમે દરેક task માટે અલગ સિક્વન્સ ડાયાગ્રામ દોરી શકો છો.

Components of Sequence diagram

Object - Class role or participants

- ઓબ્જેક્ટ ક્લાસ roles ઓબ્જેક્ટ કેવી રીતે વર્તે છે તેનું વર્ણન કરે છે.

Object : Class

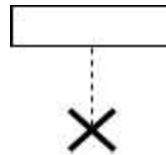
Activation

- Activation બોક્સ ટાસ્ક પૂરો કરવા માટે ઓબ્જેક્ટને જરૂરી સમય દર્શાવે છે.

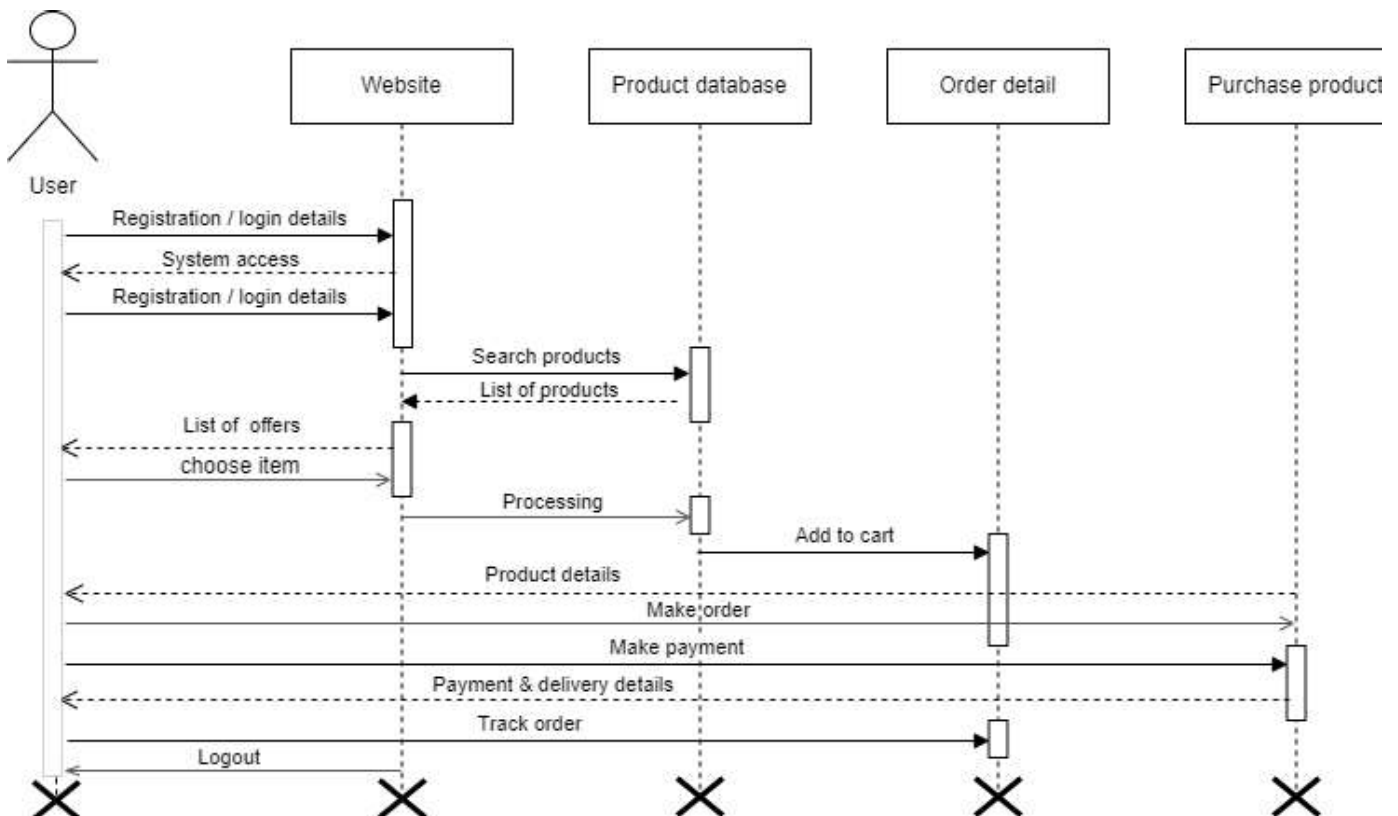


Lifeline

- જ્યારે તે ઓબ્જેક્ટની lifeline સમાપ્ત થાય છે, ત્યારે ઓબ્જેક્ટનો end દર્શાવવા માટે તેની lifeline ની અંતે X મૂકી શકો છો.



Sequence diagram for online shopping



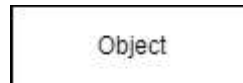
Collaboration diagram

- કોલાબોરેશન ડાયાગ્રામને કોમ્યુનિકેશન ડાયાગ્રામ તરીકે પણ ઓળખવામાં આવે છે.
- કોલાબોરેશન ડાયાગ્રામ ઓબ્જેક્ટના group ના context અને આ ઓબ્જેક્ટ વચ્ચેની interaction બંનેને express કરે છે. તે ઓબ્જેક્ટના વચ્ચેના relationships પર focus કરે છે.

Components of collaboration diagram

Object

- ઓબ્જેક્ટ naming લેવલ સાથે rectangles માં બતાવવામાં આવે છે.



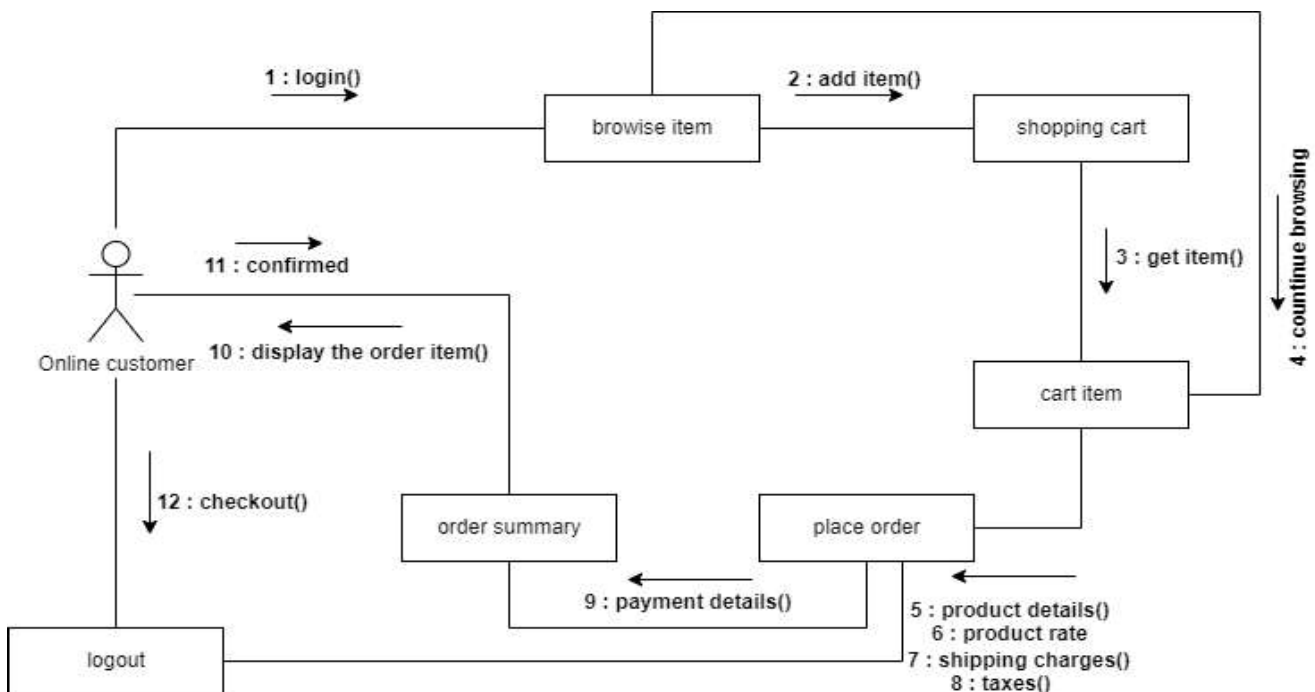
Link

- Link actor સાથે ઓબ્જેક્ટ ને connect કરે છે અને બે elements ની line નો ઉપયોગ કરીને દર્શાવવામાં આવે છે.

Message

- બે ઓબ્જેક્ટ વચ્ચેના message લિંકની બાજુ માં મૂકવામાં આવેલા લેબલવાળા arrow માં દર્શાવવામાં આવે છે. આ message એવી objects વચ્ચેનો પ્રવૃત્તિ વિશેની માહિતી આપે છે અને તેમાં sequence નંબર include થાય છે.

Collaboration diagram for online shopping

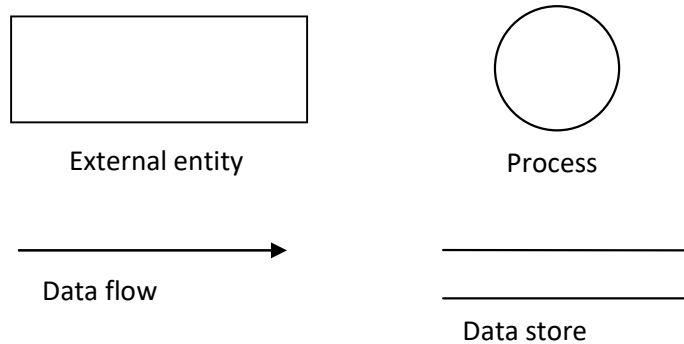


Data flow diagram

- DFD ને Bubble chart તરીકે ઓળખવામાં આવે છે. DFD એ Graphical Model છે જે System ની વિવિધ પ્રક્રિયા અથવા કાર્ય અને તે કાર્ય ની માહિતી ને Interchange કરે છે.
- દરેક Function ને એક Process તરીકે ઓળખવામાં આવે છે, જે કેટલાક Input Data ને લઈને Output Data Provide કરે છે. System ની અંદર અલગ પ્રક્રિયા પર ઉપયોગ થતા Input Data અને તેનું Output System દ્વારા Generate કરવામાં આવે છે.

Basic symbols of DFD

- DFD Model ની અંદર ખૂબ મર્યાદિત સંખ્યામાં Symbol આવેલા છે. તે Symbol દ્વારા કરવામાં આવતા કાર્ય અને તે કાર્ય ની વચ્ચે Data નો Flow દર્શાવે છે.
- **External entity:** External Entity એ એવી Physical Entity છે જે System સાથે વાતચીત કરીને અને Input Data લઈને તે Data નો ઉપયોગ કરીને Output Produce કરે છે. External Entity ને Rectangle વડે દર્શાવવામાં આવે છે. For example , Library Member
- **Function:** Function ને Circle વડે દર્શાવવામાં આવે છે. આ Symbol ને Process અથવા Bubble કહેવાય છે.
- **Data Flow:** Data Flow Symbol માં Arrow નો ઉપયોગ કરવામાં આવે છે. Data Flow બે Process અથવા બે External Entity ની વચ્ચે Data નો Flow દર્શાવે છે. અને Data Flow Arrow ની દિશામાં Process કરે છે.



- **Data store:** Data Store ને બે Parallel Line વડે દર્શાવામાં આવે છે. દરેક Data Store Process સાથે Connect હોય છે. Data Flow Arrow ની દિશા Data ને Data Store માં Read કરે છે. તે logical file દર્શાવે છે.
- **Output:** Output Symbol નો ઉપયોગ hard Copy બનાવવા માટે થાય છે.

Develop DFD model of system

- DFD Model એ System Input Data ને Transfer કરીને Final results generate કરે છે.
- Top Level DFD ને Level 0 DFD અથવા Context Diagram કહેવાય છે.
- DFD System સૌથી નીચા Level થી શરૂ કરવામાં આવે છે. અને High Level માં વધુ વિગતો ઓળખવામાં

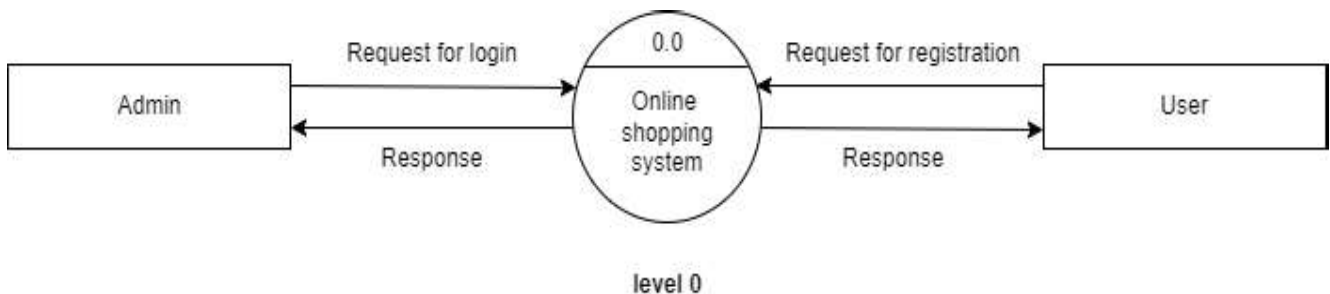
આવે છે.

- High Level DFD Model બનાવા માટે Process તેની Sub Process માં વિભાજિત થાય છે અને તે Sub Process નો Flow ને ઓળખવામાં આવે છે.
- સિસ્ટમના DFD Model બનાવા માટે, પહેલા problem ની સૌથી abstract representation પર કામ કરવું પડશે. Problem સૌથી abstract representation ને context diagram પણ કહેવામાં આવે છે. Context diagram બનાવા પછી, higher-level DFD બનાવાના હોઈ છે.

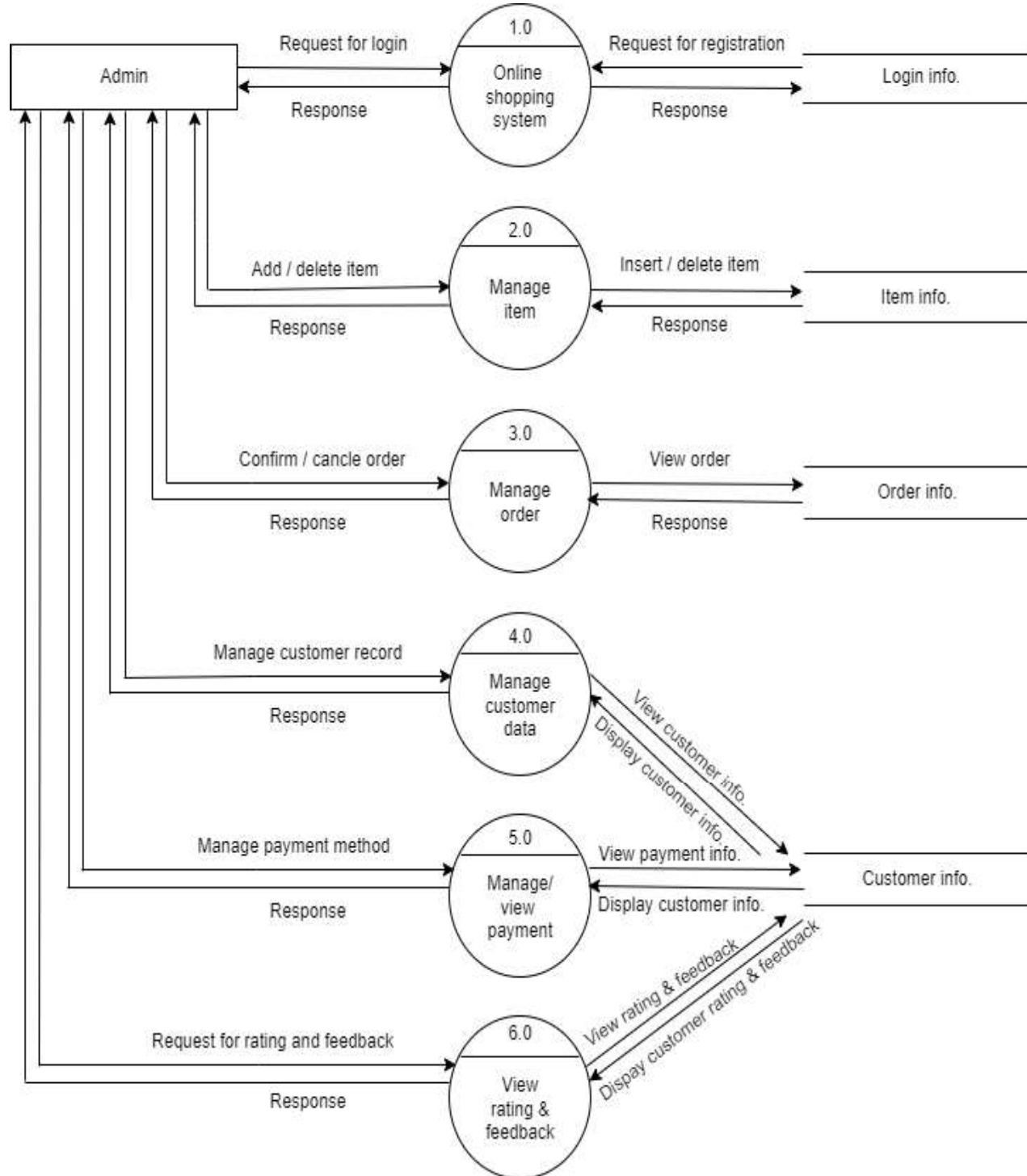
Context Diagram (Level 0)

- Context Diagram એ Abstract Data ના Flow ને દર્શાવે છે. તેમાં આખી System ને Single Bubble માં લખવામાં આવે છે.
- વિવિધ External Entity જેની સાથે System સંપર્ક કરે છે. અને તેની માહિતી અને Data નો Flow રજૂ કરવામાં આવે છે.
- System ની Data Input અને System પાસેથી મળેલું Output ને Incoming અને Output Error વડે દર્શાવામાં આવે છે.
- Context Diagram ને Level 0 તરીકે ઓળખવામાં આવે છે.

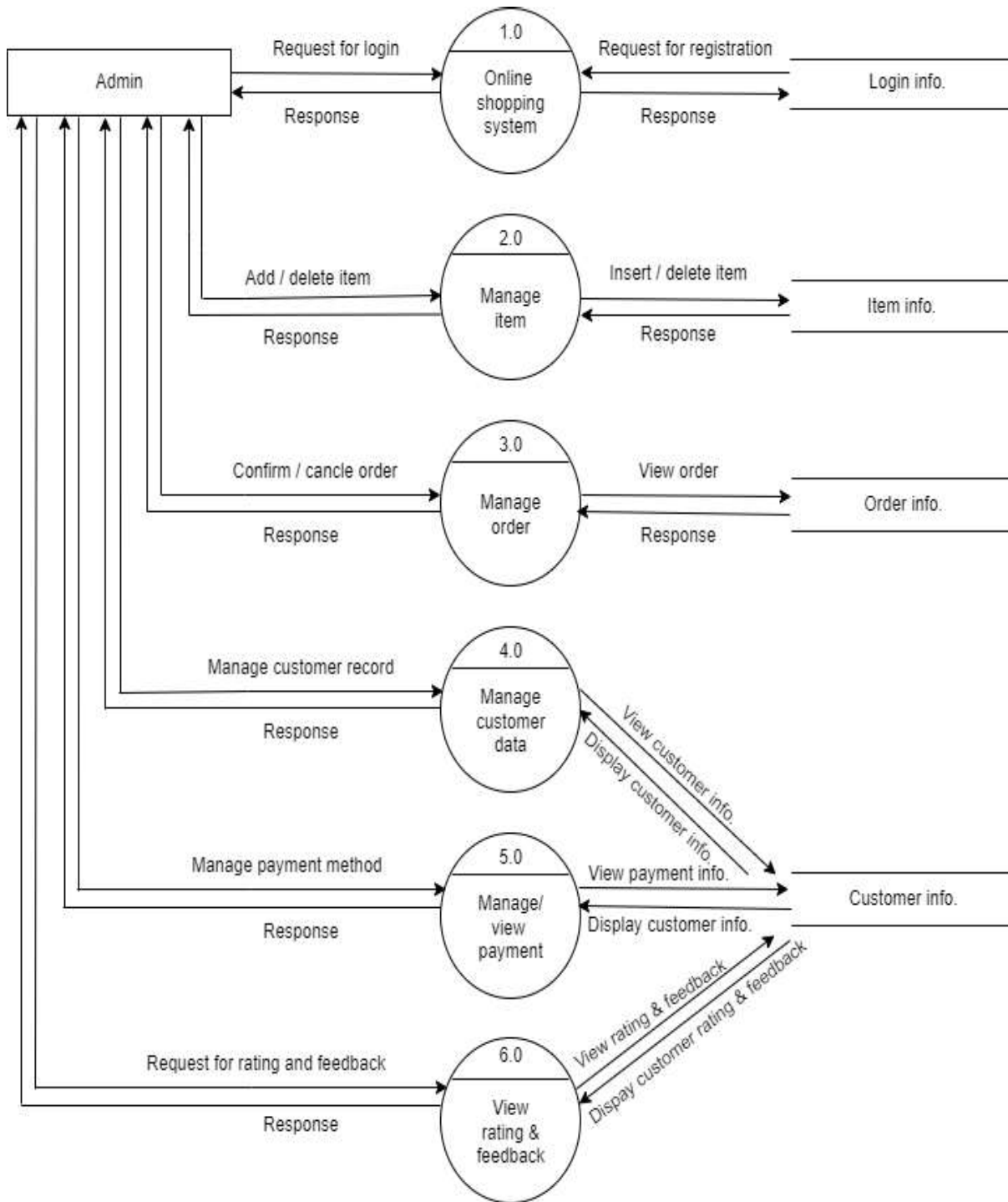
Level 0



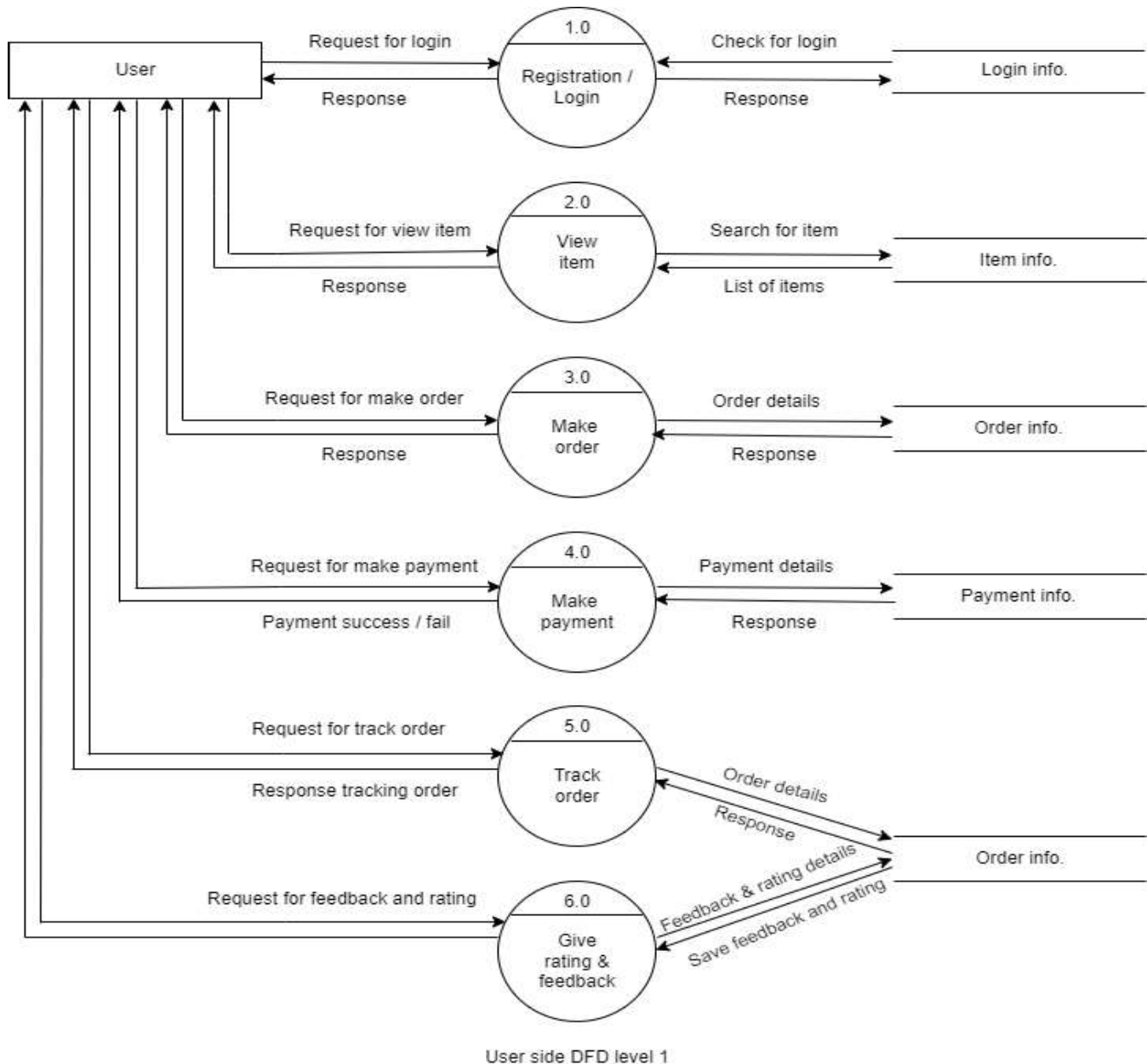
Level 1



Admin side DFD level 1



Admin side DFD level 1



Advantages of DFD

- તે Users માટે Existing System નું Knowledge આપવા માટે ફાયદાકારક છે.
- DFD System ના Components નું વિગતવાર રજૂઆત કરે છે.
- DFD ને Technical અને Non-technical માણસ સરળતાથી સમજી શકે છે.

Shortcoming (Disadvantage) of DFD Model

- DFD મોટી System માટે complex બને છે. તેને સમજવું અઘરું છે. અને Time Consuming છે.
- Data નો Flow જો સારી રીતે આપેલો ન હોય તો Programmer ને Confuse કરે છે.

- DFD માં Process માટે ઘણી શક્યતાઓ હોય છે તેથી અનેક Alternate શક્ય છે.
- DFD માં શું Input અને Output ઉત્પન્ન થાય છે. એ સ્પષ્ટ નથી.

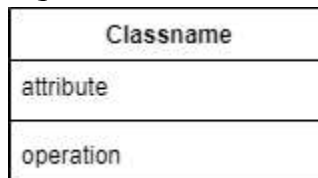
Class Diagram:

- Class Diagram નો મુખ્ય હેતુ Application ના Static View ને દર્શાવવા માટે થાય છે.
- તે System નું Structure, તેના Class, Attribute અને Object વચ્ચેના સંબંધો દર્શાવે છે.

Elements of class diagram

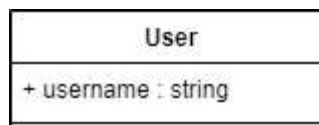
Class

- Class નું નામ upper ના section માં દેખાય છે.
- Class નું નામ meaningful હોવું જોઈએ.
- Class નું નામ capital letters થી શરૂ થવું જોઈએ અને intermediate અક્ષર કેપિટલ છે.
- Class નું નામ હંમેશા upper section માં center માં હોવું જોઈએ.
- Class નું નામ હંમેશા bold format માં હોવું જોઈએ.



Attribute

- એટ્રિબ્યુટને class ની property ઓળખવામાં આવે છે જે class ના દરેક ઓબ્જેક્ટ દ્વારા રાખવામાં આવેલ value નું describe કરે છે.
- Attribute નું નામ regular face માં હોવું જોઈએ, બોક્સમાં left align હોવું જોઈએ અને પ્રથમ અક્ષર માટે lowercase letters નો ઉપયોગ કરવો જોઈએ.
- Attribute માટેનો ડેટા પ્રકાર colon પછી લખવો જોઈએ.
- Syntax : accessModifier attributeName:dataType=defaultValue



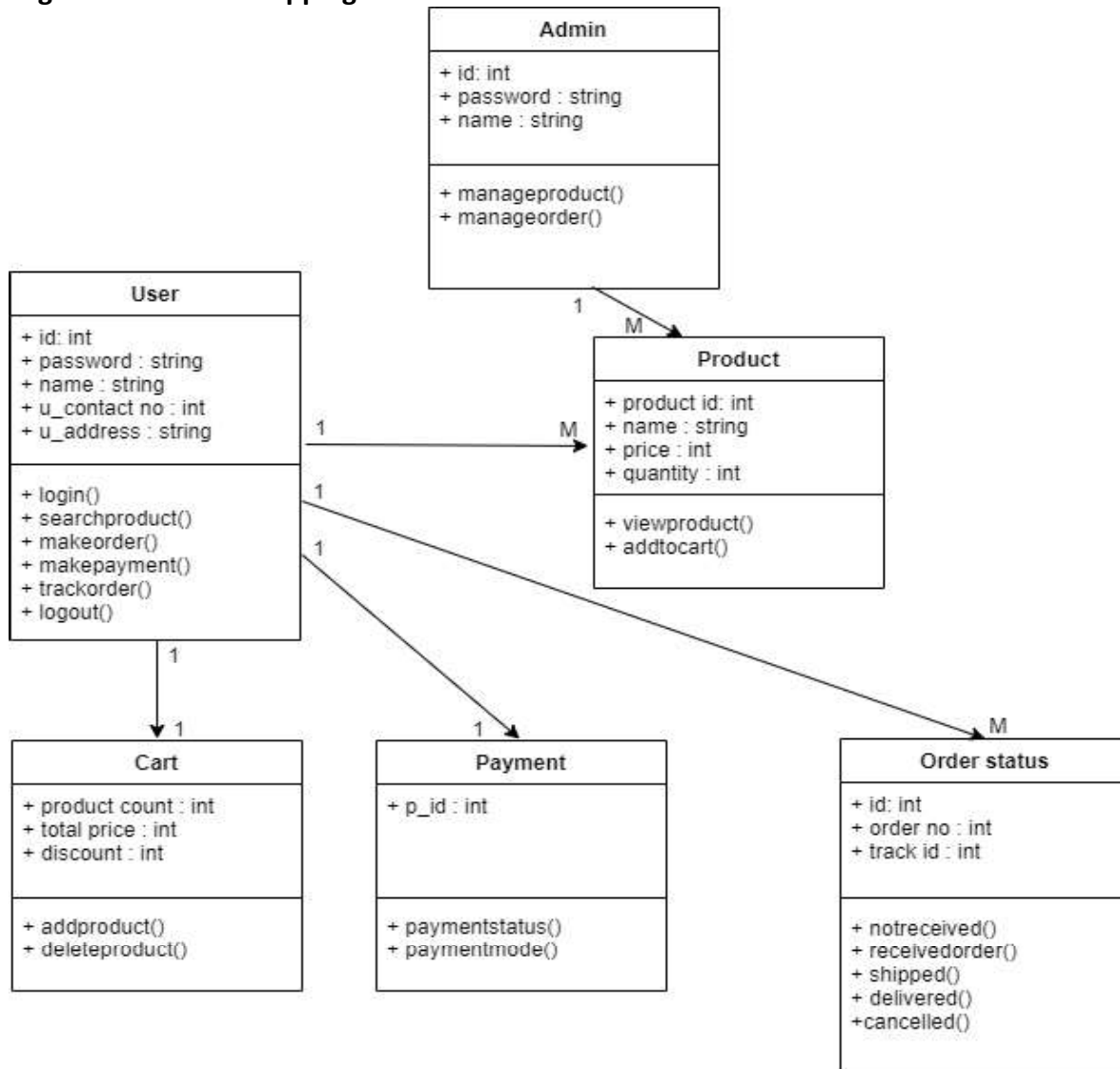
Operation

- Operation એ એક function અથવા procedure છે જે class માં ઓબ્જેક્ટ પર લાગુ થઈ શકે છે.
- Attribute નું નામ regular face માં હોવું જોઈએ, બોક્સમાં left align હોવું જોઈએ અને પ્રથમ અક્ષર માટે lowercase letters નો ઉપયોગ કરવો જોઈએ.
- Return type ની method colon પછી લખવી જોઈએ.

- Syntax : accessModifier methodName(argumentList):returnType



Class diagram for online shopping



Concept of Cohesion and Coupling

Cohesion

- Cohesion એટલે Module ની અંદર Function ના Elements ની તાકાત માપવા માટે ઉપયોગ થાય છે.
- Elements ની અંદર instructions, instructions નું group, Data-Definition અને બીજા Modules નો સમાવેશ થાય છે.

- Cohesion મતલબ કેવી રીતે સાવ નજીકથી Element ના Modules એક-બીજા સાથે સંબંધિત છે.

coincidental	Logical	Temporal	procedural	communicational	sequential	Function:
--------------	---------	----------	------------	-----------------	------------	-----------

Low —————→ High

Functional Cohesion

- તે મજબૂત Cohesion છે.
- જો બધા Module ના Elements એક task કરવા માટે related છે. ત્યારે Functional Cohesion અસ્તિત્વમાં આવે છે.
- તમામ Module Element એક Single Goal મેળવે છે.
- For example, Sort Array એ આ Module નું Example છે

Sequential Cohesion

- એવા Module ને Sequential Cohesion કહેવાય છે જ્યાં Module નો Element ક્રમ ધરાવતો હોય , જ્યાં એક Element નું Output આગળ ક્રમના element નું Input દર્શાવે છે.
- For example, એક TPS માં Input મેળવવું, Validate Output જેવા કાર્યોને એક Module ના Group માં વહેંચવામાં આવે છે.

Communicational Cohesion

- એવા Module ને Communicational Cohesion કહેવાય છે જેમાં બધા Functions સમાન માહિતીનું Structure ને Update કરે છે. E.g. Array અથવા Stack માં Function ના સમૂહ ને દર્શાવે છે.

Procedural cohesion

- જો એક Module Procedural Cohesion હોય તો , તે Module ના કાર્યો ના સમૂહ બધા Algorithm નો ભાગ હોય અને objective મેળવવા માટે ચોક્કસ પ્રક્રિયા કરવામાં આવે છે. E.g. એક Message Decode કરવા માટે Algorithm નો ઉપયોગ કરવામાં આવે છે.

Temporal Cohesion

- એવા Module ને Temporal Cohesion કહેવાય છે જેમાં તમામ કાર્યો એક જ સમય ગાળામાં થવું જોઈએ.
- Temporal Cohesion કાર્યો ના સમૂહ નું Start-Up, Shutdown Of Some Process વગેરે નો સમાવેશ કરવામાં આવે છે.

Logical Cohesion

- જેવા Module ને Logical Cohesion કહેવાય છે, જે Module ના બધા Elements સમાન કામ કરતા હોય.
- Example તરીકે Print માટે ના વિવિધ Output Report એક Single Module માં ગોઠવાયેલા હોય છે.

Coincidental Cohesion

- તે સૌથી નીચેનું Cohesion છે એવા Module ને Coincidental Cohesion કહેવાય છે કે જે, જો તે ખૂબ Loosely એકબીજા સાથે સમૂહમાં કાર્ય કરે છે , એનો અર્થ એવો થાય કે તે Module ના કાર્યો નો Randomly સમાવેશ થાય છે.
- For example, Transaction Processing System ની અંદર, Input મેળવવું, ભૂલ દર્શાવવી અને બધા

Member ને એક Module માં વહેંચવામાં આવે છે.

COUPLING

- Coupling એ બે Module વચ્ચેના relation નો આધાર અથવા બે Modules વચ્ચેના ક્રિયા-પ્રતિક્રિયા ની Degree નું માપ છે.
- તે System માં Module વચ્ચેના strengths નો ઉલ્લેખ કરે છે. તે કેવી રીતે બે Module નજીક થી ઓળખ શકાય છે અને કેવી રીતે તે સ્વતંત્ર છે તે સૂચવે છે.
- Modules વધારે સ્વતંત્ર બની જાય છે, જ્યારે coupling વધે છે અને Loose Coupling તેની સ્વતંત્રતાને ઓછી કરે છે. તે System Development માટે સારી છે.
- જો બે Modules વધુ Amount માં Data ને બદલતા હોય તો તે વધુ સ્વતંત્ર હોય છે.
- બે Module વચ્ચેનું High Coupling એ System ને સમજવી Difficult બનાવે છે અને system બનાવવા માટેના પ્રયત્નો વધારે છે.
- તેની Low Coupling Best છે.

Data	Stamp	Control	Common	Content
Low				High

Data coupling

- બે Module data coupling હોય છે. જો તે Parameter મારફતે Communication કરે.
- For Example, પ્રાથમિક માહિતી બે Module વચ્ચેના Parameter તરીકે Pass થાય છે. E .g . Integer, Float, Character etc...
- તે Lowest Coupling છે અને Software વિકાસ માટે Best છે.

Stamp coupling

- બે Module Stamp Couple હોય છે. જો તે સંયુક્ત માહિતી (Composite Data) સાથે Communicate કરે છે. જેવી કે C ની Structure અને Pascal ની અંદરનો Record.

Control coupling

- Control Coupling બે Module વચ્ચે આવેલું હોય છે. જો એક Module નો Data નો ઉપયોગ Direct સૂચનાઓ , Order ના અમલ અન્ય module માં કરવા માટે વપરાય છે.
- Control Coupling નું Example એક Module માં Flag ને Set કરવો અને બીજા Module માં Test કરવો.

Common coupling

- જો બે Module Global Data Item દ્વારા માહિતીને Share કરે છે. તો તેને Common Coupled કહેવાય છે .

Content coupling

- Content Coupling બે Module વચ્ચે હોય છે, જો તે Code ને Share કરે. તે એક Module માંથી બીજા Module માં Jump કરે છે.
- For e.g. અન્ય Module માં એક Module માંથી Branch condition.