FLIPKART

A Software Engineering Lab report submitted in partial fulfilment of the requirement for the Award of the Degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

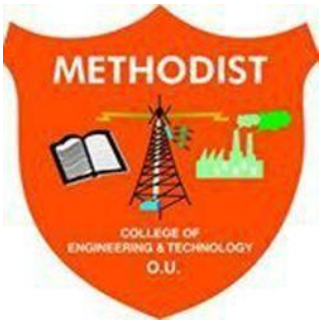
TAUSEEF ALI KHAN (160719733003)

SANTOSH NAGA MANIDEEP.B (160719733002)

CH UPENDRA (160719733019)

Under the Guidance of

Er. Sandeep Ravikanti, Assistant Professor, Dept. of CSE



Department of Computer Science and Engineering

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

2021-2022

METHODIST
College of Engineering & Technology
(Approved by AICTE, New-Delhi & Affiliated to Osmania University)
College Code : 1607

Dr. Prabhu G Benakop
B.E., M.E., Ph.D.
SM., IEEE, LMISTE, LMISOI
Principal

VISION

To produce ethical, socially conscious and innovative professionals who would contribute to sustainable technological development of the society.

MISSION

- To impart quality engineering education with latest technological developments and interdisciplinary skills to make students succeed in professional practice.
- To encourage research culture among faculty and students by establishing state of art laboratories and exposing them to modern industrial and organizational practices.
- To inculcate humane qualities like environmental consciousness, leadership, social values, professional ethics and engage in independent and lifelong learning for sustainable contribution to the society.

PRINCIPAL
PRINCIPAL
METHODIST COLLEGE OF ENGG.& TECH.
King Koti Road, Abids, Hyderabad.

King Koti Road, Abids
Hyderabad - 500 001. T.S. India
Ph : 040-24753445, 24755999
www.methodist.edu.in

METHODIST COLLEGE OF ENGINEERING & TECHNOLOGY

Affiliated to Osmania University, - College Code – 1607

Department of Computer Science & Engineering

VISION

To become a leader in providing Computer Science & Engineering education with emphasis on knowledge and innovation.

MISSION

- To offer flexible programs of study with collaborations to suit industry needs
- To provide quality education and training through novel pedagogical practices
- To expedite high performance of excellence in teaching, research and innovations.
- To impart moral, ethical valued education with social responsibility.

Head of the Department
Department of CSE
Methodist College of Engg & Tech
Abids, Hyderabad.

METHODIST COLLEGE OF ENGINEERING & TECHNOLOGY

Affiliated to Osmania University, - College Code – 1607

Department of Computer Science & Engineering

PROGRAM OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in

Program Specific Outcomes

At the end of 4 years, Compute Science and Engineering graduates at MCET will be able to:

PSO1: Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.

PSO2: Develop software applications with open-ended programming environments.

PSO3: Design and develop solutions by following standard software engineering principles and implement by using suitable programming languages and platforms

METHODIST COLLEGE OF ENGINEERING & TECHNOLOGY

Affiliated to Osmania University, - College Code – 1607

Department of Computer Science & Engineering

Program Educational Objectives

Graduates of Compute Science and Engineering at Methodist College of Engineering and Technology will be able to:

PEO1: Apply technical concepts, Analyze, Synthesize data to Design and create novel products and solutions for the real life problems.

PEO2: Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.

PEO3: Promote collaborative learning and spirit of team work through multidisciplinary projects

PEO4: Engage in life-long learning and develop entrepreneurial skills.

Head of the Department
Department of CSE
Methodist College of Engg. & Tech
Abids, Hyderabad.

METHODIST
COLLEGE OF ENGINEERING & TECHNOLOGY
(An UGC-AUTONOMOUS INSTITUTION)
Accredited by NAAC with A+ and NBA
Affliated to Osmania University & Approved by AICTE
Estd : 2008

King Koti, Abids, Hyderabad-500001,

Department of Computer Science and Engineering

SOFTWARE ENGINEERING LAB (PC 532 CS) A.Y 2021-2022

This is to certify that this Software Engineering Lab report entitled "**FLIPKART",** being submitted *by* **TAUSEEF ALI KHAN (160719733003), SANTOSH NAGA MANIDEEP.B (160719733002), CH UPENDRA (160719733019)**, submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering, during the academic year 2021-2022, is a bonafide record of work carried out by them.

INTERNAL **EXTERNAL** **HOD**

King Koti, Abids, Hyderabad-500001,

METHODIST
COLLEGE OF ENGINEERING & TECHNOLOGY
(An UGC-AUTONOMOUS INSTITUTION)
Accredited by NAAC with A+ and NBA
Affliated to Osmania University & Approved by AICTE
Estd : 2008

Department of Computer Science and Engineering

DECLARATION BY THE CANDIDATES

We, **TAUSEEF ALI KHAN (160719733003), SANTOSH NAGA MANIDEEP.B (160719733002), CH UPENDRA (160719733019)** students of Methodist College of Engineering and Technology, pursuing Bachelor's degree in Computer Science and Engineering, hereby declare that Software Engineering Lab report entitled "**FLIPKART",** carried out under the guidance of **Mr. Sandeep Ravikanti** submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science. This work is carried out by us and the references have been taking from various digital resources for report preparation.

TAUSEEF ALI KHAN (160719733003)
SANTOSH NAGA MANIDEEP.B (160719733002)
CH UPENDRA (160719733019)

King Koti, Abids, Hyderabad-500001,

METHODIST
COLLEGE OF ENGINEERING & TECHNOLOGY
(An UGC-AUTONOMOUS INSTITUTION)
Accredited by NAAC with A+ and NBA
Estd : 2008 Affliated to Osmania University & Approved by AICTE

Department of Computer Science and Engineering

CERTIFICATE BY THE LAB INCHARGE

This is to certify that this Software Engineering Lab report entitled "**FLIPKART",** being submitted *by* **TAUSEEF ALI KHAN (160719733003), SANTOSH NAGA MANIDEEP.B (160719733002), CH UPENDRA (160719733019)**, submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering, during the academic year 2021-2022, is a bonafide record of work carried out by them.

Er. Sandeep Ravikanti
AssistantProfessor,
Dept. of CSE

King Koti, Abids, Hyderabad-500001,

Department of Computer Science and Engineering

METHODIST
COLLEGE OF ENGINEERING & TECHNOLOGY
(An UGC-AUTONOMOUS INSTITUTION)
Accredited by NAAC with A+ and NBA
Affliated to Osmania University & Approved by AICTE
Estd : 2008

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that this Software Engineering Lab report entitled "**FLIPKART**" *by*

TAUSEEF ALI KHAN (160719733001), SANTOSH NAGA MANIDEEP.B (160719733002), CH UPENDRA (160719733019), submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2021-2022, is a bonafide record of work carried out by them.

Dr. P. Lavanya,
Professor &
Head of the Department

Overview of StarUML™5.0

StarUML™ is a software modeling platform that supports UML (Unified Modeling Language). It is based on UML version 1.4 and provides eleven different types of diagram, and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept. StarUML™ excels in customizability to the user's environment and has a high extensibility in its functionality. Using StarUML™, one of the top leading software modeling tools, will guarantee to maximize the productivity and quality of your software projects.

StarUML™ provides maximum customization to the user's environment by offering customizing variables that can be applied in the user's software development methodology, project platform, and language

Software architecture is a critical process that can reach 10 years or more into the future. The intention of the OMG (Object Management Group) is to use MDA (Model Driven Architecture) technology to create platform independent models and allow automatic acquisition of platform dependent models or codes from platform independent models. StarUML™ truly complies with UML 1.4standards, UML 2.0 notation and provides the UML Profile concept, allowing creation of platform independent models. Users can easily obtain their end products through simple template document.

StarUML™ provides excellent extensibility and flexibility. It provides Add-In frameworks for extending the functionality of the tool. It is designed to allow access to all functions of the model/meta-model and tool through COM Automation, and it provides extension of menu and option items. Also, user scan create their own approaches and frameworks according to their methodologies. The tool can also be integrated with any external tools.

StarUML™ has the following features.

Feature	Description
Accurate UML standard model	StarUML™ strictly adheres to the UML standard specification specified by the OMG for software modeling. Considering the fact that the results of design information can reach 10 years or more into the future, dependence on vendor-specific irregular UML syntax and semantics can be quite risky. StarUML™ maximizes itself to order UML 1.4 standard and meaning, and it accepts UML 2.0 notation on the basis of robust meta model.
Open software model format	Unlike many existing products that manage their own legacy format models inefficiently, StarUML™ manages all files in the standard XML format. Codes written in easy-to-read structures and their formats can be changed conveniently by using the XML parser. Given the fact that XML is a world standard, this is certainly a great advantage, ensuring that the software models remain useful for more than a decade.
True MDA support	StarUML™ truly supports UML Profile. This maximizes extensibility of UML, making modeling of applications possible even in areas like finance, defense, e-business, insurance, and aeronautics. Truly Platform Independent Models (PIM) can be created, and Platform Specific Model (PSM) and executable codes can be automatically generated in any way.

Applicability of methodologies and platforms	StarUML™ manipulates the approach concept, creating environments that adapt to any methodologies/processes. Not only the application framework models for platforms like .NET and J2EE, but also basic structures of software models (e.g. 4+1 view-model, etc.) can be defined easily
Excellent extensibility	All functions of the StarUML™ tools are automated according to Microsoft COM. Any language which supports COM (Visual Basic Script, Java Script, VB, Delphi, C++, C#, VB.NET, Python, etc.) can be used to control StarUML™ or develop integrated Add-In elements.
Software model verification function	Users can make many mistakes during software modeling. Such mistakes can be very costly if left uncorrected until the final coding stage. In order to prevent this problem, StarUML™ automatically verifies the software model developed by the user, facilitating early discovery of errors, and allowing more faultless and complete software development.
Useful Add-Ins	StarUML™ includes many useful Add-Ins with various functionalities: it generates source codes in programming languages and converts source codes into models, imports Rational Rose files, exchanges model ing information with other tools using XMI, and supports design patterns. These Add-Ins offer additional reusability, productivity, flexibility and interoperability for the modeling information.

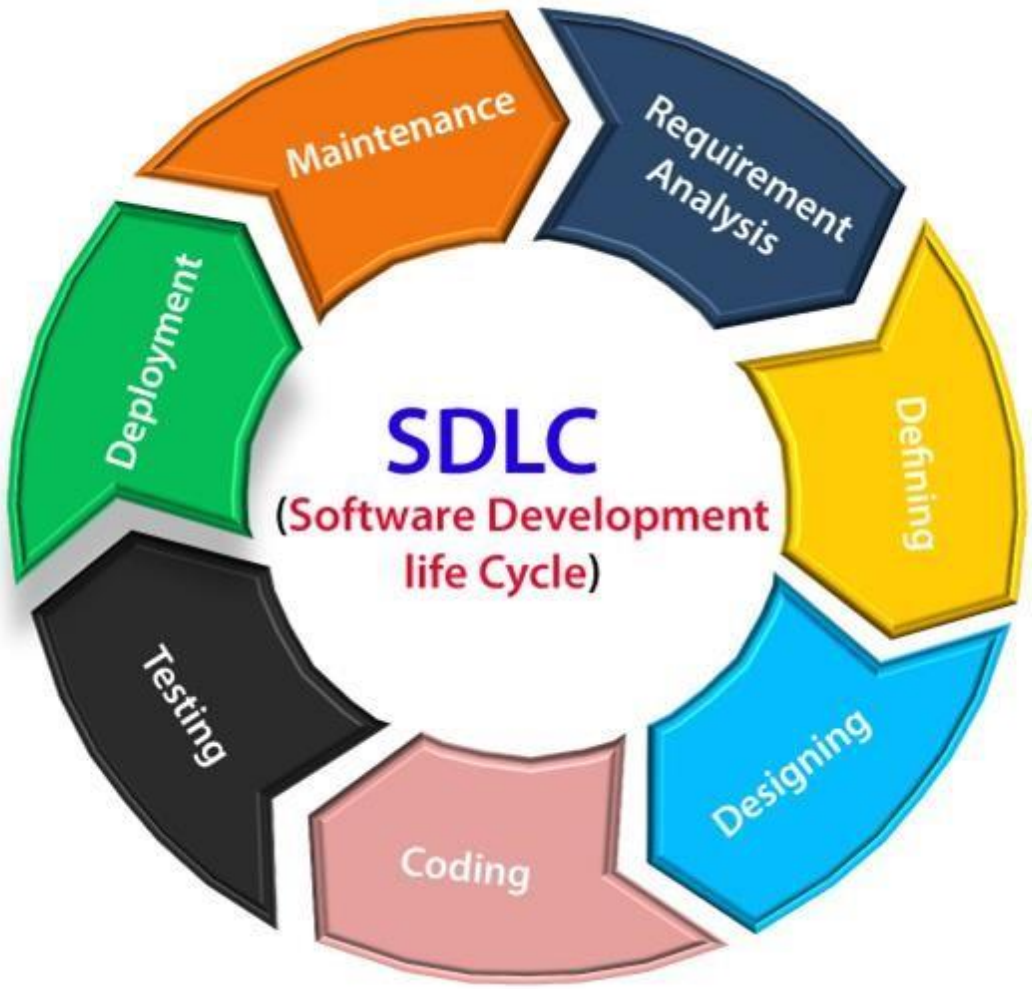
The following are the minimum System Requirements for running StarUML™

- Intel(R) Pentium(R) 233MHz or higher
- Windows(R) 2000, Windows XP™, or higher
- Microsoft(R) Internet Explorer 5.0 or
- higher 128 MB RAM (256MB recommended)
- 110 MB hard disc space (150MB space recommended) CD-ROM drive
- SVGA or higher resolution monitor (1024x768 recommended) Mouse or other pointing device

Software Development Life Cycle (SDLC)

A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

SDLC framework includes the following steps:



INTRODUCTION TO UML

Contents:

1. Importance of Modeling
2. Principles of Modeling
3. Object Oriented Modeling
4. Conceptual Model of the UML
5. Architecture
6. Software Development Life Cycle

1. Importance of Modeling:

Why do we model?

A model is a simplification at some level of abstraction We build models to better understand the systems we are developing.

- To help us visualize
- To specify structure or behavior
- To provide template for building system To
- document decisions we have made

2. Principles of Modeling:

- The models we choose have a profound influence on the solution we provide
- Every model may be expressed at different levels of abstraction
- The best models are connected to reality
- No single model is sufficient, a set of models is needed to solve any nontrivial system

UML is a visual modeling language

"A picture is worth a thousand words." - old saying Unified Modeling Language:

"A language provides a vocabulary and the rules for combining words [...] for the purpose of communication. A modeling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system. A modeling language such as the UML is thus a standard language for software blueprints." **Usages of UML:**

UML is used in the course to i. document designs • design patterns / frameworks

ii. represent different views/aspects of design – visualize and construct designs • static / dynamic / deployment / modular aspects

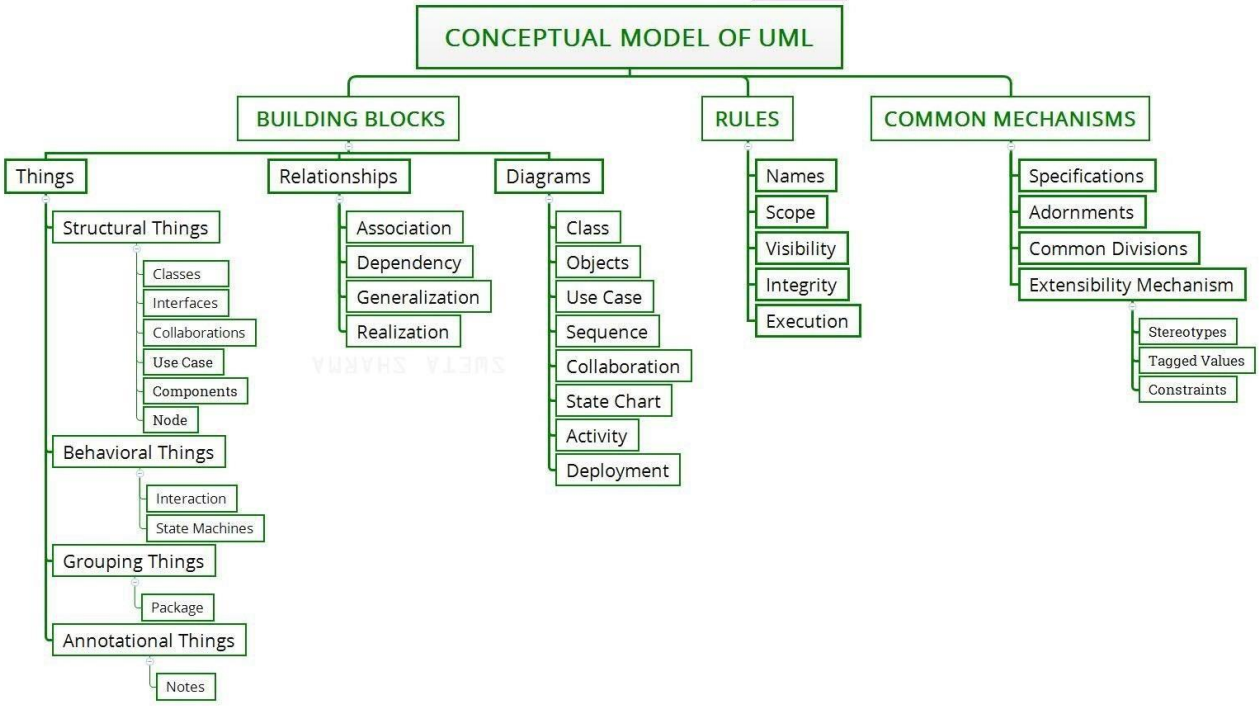
iii. provide a next-to-precise, common, language–specify visually • for the benefit of analysis, discussion, comprehension...

abstraction takes precedence over precision!

- 20/80 rule
- aim is overview and comprehension; not execution

Object Oriented Modeling:

Traditionally two approaches to modeling a software system Algorithmically – becomes hard to focus on as the requirements change Object-oriented – models more closely real world entities



To understand the UML, you need to form a conceptual model of the language, and this requires learning three major elements: the UML's basic building blocks, the rules that dictate how those building blocks may be put together, and some common mechanisms that apply throughout the UML. Once you have grasped these ideas, you will be able to read UML models and create some basic ones. As you gain more experience in applying the UML, you can build on this conceptual model, using more advanced features of the language.

Building Blocks of the UML

The vocabulary of the UML encompasses three kinds of building blocks:

1. Things
2. Relationships
3. Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things. **Things in the UML**

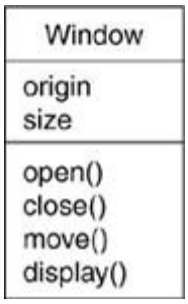
There are four kinds of things in the UML:

1. Structural things
2. Behavioral things
3. Grouping things
4. Annotational things

These things are the basic object-oriented building blocks of the UML. You use them to write wellformed models. **Structural Things**

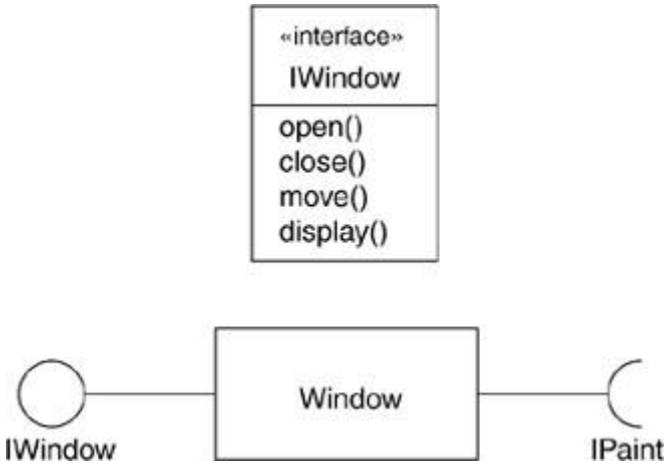
Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. Collectively, the structural things are called *classifiers*. A *class* is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations, as in Figure 2-1.

Figure 2-1. Classes



An *interface* is a collection of operations that specify a service of a class or component. An interface therefore describes the externally visible behavior of that element. An interface might represent the complete behavior of a class or component or only a part of that behavior. An interface defines a set of operation specifications (that is, their signatures) but never a set of operation implementations. The declaration of an interface looks like a class with the keyword
«interface» above the name; attributes are not relevant, except sometimes to show constants. An interface rarely stands alone, however. An interface provided by a class to the outside world is shown as a small circle attached to the class box by a line. An interface required by a class from some other class is shown as a small semicircle attached to the class box by a line, as in Figure 2- 2.

Figure 2-2. Interfaces



A *collaboration* defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. Collaborations have structural, as well as behavioral, dimensions. A given class or object might participate in several collaborations. These collaborations therefore represent the implementation of patterns that make up a system. Graphically, a collaboration is rendered as an ellipse with dashed lines, sometimes including only its name, as in Figure 2-3.

Figure 2-3. Collaborations

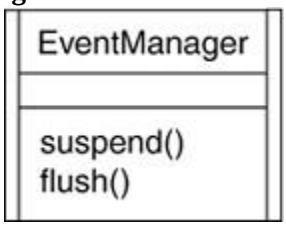
A *use case* is a description of sequences of actions that a system performs that yield observable results of value to a particular actor. A use case is used to structure the behavioral things in a model. A use case is realized by a collaboration. Graphically, a use case is rendered as an ellipse with solid lines, usually including only its name, as in Figure 2-4.

Figure 2-4. Use Cases

The remaining three things active classes, components, and nodes are all class-like, meaning they also describe sets of entities that share the same attributes, operations, relationships, and semantics. However, these three are different enough and are necessary for modeling certain aspects of an object-oriented system, so they warrant special treatment.

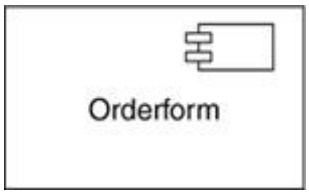
An *active class* is a class whose objects own one or more processes or threads and therefore can initiate control activity. An active class is just like a class except that its objects represent elements whose behavior is concurrent with other elements. Graphically, an active class is rendered as a class with double lines on the left and right; it usually includes its name, attributes, and operations, as in Figure 2-5.

Figure 2-5. Active Classes



A component is a modular part of the system design that hides its implementation behind a set of external interfaces. Within a system, components sharing the same interfaces can be substituted while preserving the same logical behavior. The implementation of a component can be expressed by wiring together parts and connectors; the parts can include smaller components. Graphically, a component is rendered like a class with a special icon in the upper right corner, as in Figure 2-6.

Figure 2-6. Components



The remaining two elements artifacts and nodes are also different. They represent physical things, whereas the previous five things represent conceptual or logical things.

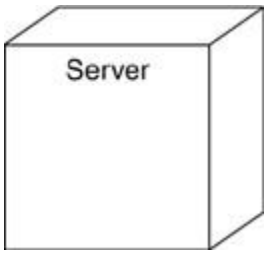
An *artifact* is a physical and replaceable part of a system that contains physical information ("bits"). In a system, you'll encounter different kinds of deployment artifacts, such as source code files, executables, and scripts. An artifact typically represents the physical packaging of source or run-time information.

Graphically, an artifact is rendered as a rectangle with the keyword «artifact» above the name, as in Figure 2-7.

Figure 2-7. Artifacts

A *node* is a physical element that exists at run time and represents a computational resource, generally having at least some memory and, often, processing capability. A set of components may reside on a node and may also migrate from node to node. Graphically, a node is rendered as a cube, usually including only its name, as in Figure 2-8.

Figure 2-8. Nodes



These elements, classes, interfaces, collaborations, use cases, active classes, components, artifacts, and nodes are the basic structural things that you may include in a UML model. There are also variations on these, such as actors, signals, and utilities (kinds of classes); processes and threads (kinds of active classes); and applications, documents, files, libraries, pages, and tables (kinds of artifacts). **Behavioral Things**

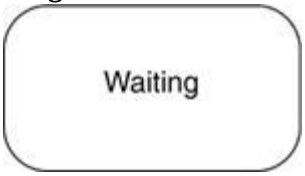
Behavioral things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space. In all, there are three primary kinds of behavioral things.

First, an *interaction* is a behavior that comprises a set of messages exchanged among a set of objects or roles within a particular context to accomplish a specific purpose. The behavior of a society of objects or of an individual operation may be specified with an interaction. An interaction involves a number of other elements, including messages, actions, and connectors (the connection between objects). Graphically, a message is rendered as a directed line, almost always including the name of its operation, as in Figure 2-9.

Figure 2-9. Messages

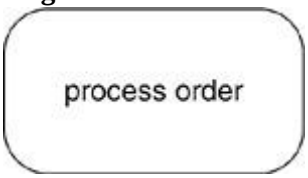
Second, a *state machine* is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its responses to those events. The behavior of an individual class or a collaboration of classes may be specified with a state machine. A state machine involves a number of other elements, including states, transitions (the flow from state to state), events (things that trigger a transition), and activities (the response to a transition). Graphically, a state is rendered as a rounded rectangle, usually including its name and its sub-states, if any, as in Figure 2-10.

Figure 2-10. States



Third, an activity is a behavior that specifies the sequence of steps a computational process performs. In an interaction, the focus is on the set of objects that interact. In a state machine, the focus is on the life cycle of one object at a time. In an activity, the focus is on the flows among steps without regard to which object performs each step. A step of an activity is called an *action*. Graphically, an action is rendered as a rounded rectangle with a name indicating its purpose. States and actions are distinguished by their different contexts.

Figure 2-11. Actions



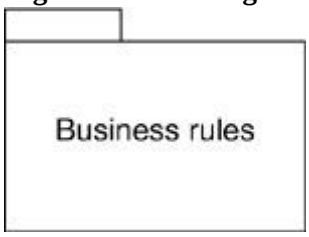
These three elements interactions, state machines, and activities are the basic behavioral things that you may include in a UML model. Semantically, these elements are usually connected to various structural elements, primarily classes, collaborations, and objects.

Grouping Things

Grouping things are the organizational parts of UML models. These are the boxes into which a model can be decomposed. There is one primary kind of grouping thing, namely, packages.

A *package* is a general-purpose mechanism for organizing the design itself, as opposed to classes, which organize implementation constructs. Structural things, behavioral things, and even other grouping things may be placed in a package. Unlike components (which exist at run time), a package is purely conceptual (meaning that it exists only at development time). Graphically, a package is rendered as a tabbed folder, usually including only its name and, sometimes, its contents, as in Figure 2-12.

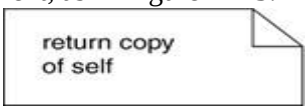
Figure 2-12. Packages



Packages are the basic grouping things with which you may organize a UML model. There are also variations, such as frameworks, models, and subsystems (kinds of packages).

Annotational Things

Annotational things are the explanatory parts of UML models. These are the comments you may apply to describe, illuminate, and remark about any element in a model. There is one primary kind of annotational thing, called a note. A *note* is simply a symbol for rendering constraints and comments attached to an element or a collection of elements. Graphically, a note is rendered as a rectangle with a dog-eared corner, together with a textual or graphical comment, as in Figure 2- 13. **Figure 2-13. Notes**



This element is the one basic annotational thing you may include in a UML model. You'll typically use notes to adorn your diagrams with constraints or comments that are best expressed in informal or formal text. There are also variations on this element, such as requirements (which specify some desired behavior from the perspective of outside the model).

Relationships in the UML

There are four kinds of relationships in the UML:

1. Dependency
2. Association
3. Generalization 4. Realization

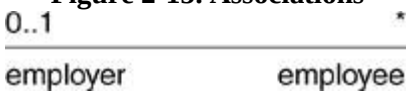
These relationships are the basic relational building blocks of the UML. You use them to write wellformed models.

First, a *dependency* is a semantic relationship between two model elements in which a change to one element (the independent one) may affect the semantics of the other element (the dependent one). Graphically, a dependency is rendered as a dashed line, possibly directed, and occasionally including a label, as in Figure 2-14.

Figure 2-14. Dependencies

Second, an *association* is a structural relationship among classes that describes a set of links, a link being a connection among objects that are instances of the classes. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts. Graphically, an association is rendered as a solid line, possibly directed, occasionally including a label, and often containing other adornments, such as multiplicity and end names, as in Figure 2- 15.

Figure 2-15. Associations



Third, a *generalization* is a specialization/generalization relationship in which the specialized element (the child) builds on the specification of the generalized element (the parent). The child shares the structure and the behavior of the parent. Graphically, a generalization relationship is rendered as a solid line with a hollow arrowhead pointing to the parent, as in Figure 2-16.

Figure 2-16. Generalizations

Fourth, a *realization* is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out. You'll encounter realization relationships in two places: between interfaces and the classes or components that realize them, and between use cases and the collaborations that realize them. Graphically, a realization relationship is rendered as a cross between a generalization and a dependency relationship, as in Figure 2-17.

Figure 2-17. Realizations

These four elements are the basic relational things you may include in a UML model. There are also variations on these four, such as refinement, trace, include, and extend.

Diagrams in the UML

A *diagram* is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and paths (relationships). You draw diagrams to visualize a system from different perspectives, so a diagram is a projection into a system. For all but the most trivial systems, a diagram represents an elided view of the elements that make up a system. The same element may appear in all diagrams, only a few diagrams (the most common case), or in no diagrams at all (a very rare case). In theory, a diagram may contain any combination of things and relationships. In practice, however, a small number of common combinations arise, which are consistent with the five most useful views that comprise the architecture of a software- intensive system. For this reason, the UML includes thirteen kinds of diagrams:

1. Class diagram
2. Object diagram
3. Component diagram
4. Composite structure diagram
5. Use case diagram
6. Sequence diagram

7. Communication diagram
8. State diagram
9. Activity diagram
10. Deployment diagram
11. Package diagram
12. Timing diagram
13. Interaction overview diagram

STARUML TOOL

GROUPING ELEMENTS PROVIDED IN STARUML

Grouping Element	Description
Model	Model expresses the physical system for specific purposes (aspects). For example, it can express a specific aspect of the system (e.g. analysis aspect, design aspect, user aspect, etc.).
Subsystem	Subsystem groups the elements that specify the entire physical system or parts of it.
Package	Package logically groups and manages model elements. It is an extremely generalized element that can be used in any way for organizing elements.

Procedure for Creating New Diagram:

1. Select from the model explorer or diagram area an element to contain the new diagram.
2. Right-click and select the **[Add Diagram]** menu. A new diagram will be created when selection is made for the diagram type.

Types of Diagrams Available

Diagram Type	Description
Class Diagram	Class Diagram is a visual expression of various static relations of class-related elements. Class Diagram can contain not only classes but also interfaces, enumerations, packages, various relations, instances, and their links.
Use Case Diagram	Use Case Diagram is an expression of relations between the use cases in a specific system or object and the external actors. Use Case expresses the functions of the system and how the system functions interact with the external actors.
Sequence Diagram	Sequence Diagram expresses the interactions of instances. It is a direct expression of the InteractionInstanceSet, which is a set of the stimuli exchanged between the instances within a CollaborationInstanceSet. While Sequence Role Diagram is a ClassifierRole-oriented expression, Sequence Diagram is an Instance-oriented expression.
Sequence Diagram (Role)	Sequence Role Diagram expresses the interactions of the role concepts. It is a direct expression of the Interaction, which is a set of the messages exchanged between the ClassifierRoles within a Collaboration. While Sequence Diagram is an Instance-oriented expression, Sequence Role Diagram is a ClassifierRole-oriented expression.

Collaboration Diagram	Collaboration Diagram expresses the collaboration between instances. It is a direct expression of the collaboration model of the instances within a CollaborationInstanceSet. While Collaboration Role Diagram is a ClassifierRole-oriented expression, Collaboration Diagram is an Instance-oriented expression.
Collaboration Diagram (Role)	Collaboration Role Diagram expresses the collaboration between the role concepts. It is a direct expression of the collaboration model of the ClassifierRoles within a Collaboration. While Collaboration Diagram is an Instanceoriented expression, Collaboration Role Diagram is a ClassifierRole-oriented expression.
Statechart Diagram	Statechart Diagram expresses the static behaviors of a specific object through states and their transitions. Although Statechart Diagram is generally used to express the behaviors for instances of classes, it can also be used to express behaviors of other elements.
Activity Diagram	Activity Diagram is a special form of Statechart Diagram that is suitable for expressing the activity execution flow. Activity Diagram is commonly used for expressing workflow, and it is frequently used for objects like classes, packages, and operations.
Component Diagram	Component Diagram expresses the dependency between the software components. The elements that constitute software components and the elements that implement those components can all be expressed by Component Diagram.
Deployment Diagram	Deployment Diagram expresses the hardware elements of the physical computer and devices and the software components, processes and objects that are assigned to them.
Composite Structure Diagram	Composite Structure Diagram is a diagram to express internal structure of Classifier. It is included in interaction point with other parts of system.

Property Types

Property Type	Description
Name	Indicates the name of the model element.
Stereotype	Indicates the stereotype for the model element.
TypeExpression	Indicates the expression for special type.
String	Indicates string.
Boolean	Indicates True or False.

Enumeration	Selects one of the various literals.
Reference	Indicates a specific element.
Collection	Indicates multiple elements (editable through the collection editor).

SAVING DIAGRAM AS IMAGE FILE

Diagrams can be saved as image files. StarUML™ supports these image formats: JPEG (.jpg, .jpeg), bitmap (.bmp), metafile (.wmf), and extended metafile (.emf).

Procedure for Saving Diagram as Image:

1. Make a diagram to save as image the active diagram.
2. Select [File] -> [Export Diagram…] from the main menu.
3. At the Save dialog box, enter the file name, select the file format, and then click the [Save] button. Note

Modeling with UseCase Diagram

The following elements are available in a usecase diagram.

- Actor
- UseCase
- Association
- Derected Association
- Generalization
- Dependency
- Include
- Extend
- System Boundary
- Package

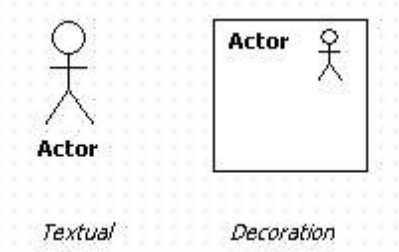
ACTOR

Semantics

An actor defines a coherent set of roles that users of an entity can play when interacting with the entity. An actor may be considered to play a separate role with regard to each use case with which it communicates.

Procedure for creating Actor

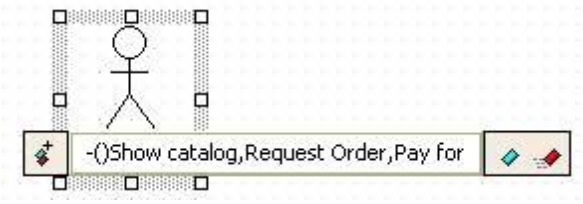
In order to create Actor, click [Toolbox] -> [UseCase] -> [Actor] button and click the position where to place Actor. Actor is shown in the form of stick man or rectangle with icon, that is decoration view. To display actor in decoration view, select [Format] -> [Stereotype Display] -> [Decoration] menu item or select [Decoration] item in [] combo button on toolbar.



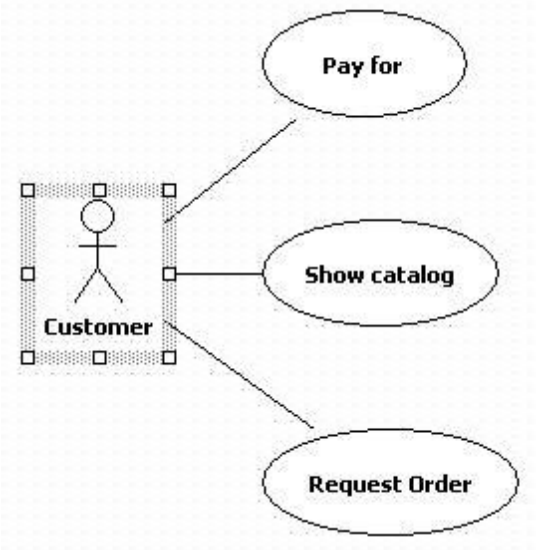
Procedure for creating multiple UseCases used by Actor at once

In order to create multiple UseCases related to Actor at once, use shortcut creation syntax of Actor.

1. At the Actor's quick dialog, enter UseCase's name after "-()" string. To create multiple UseCases, enter same but separate UseCase's name by "," character.



2. And press [Enter] key. Several UseCases associated with the Actor are created and arranged vertically.



USECASE

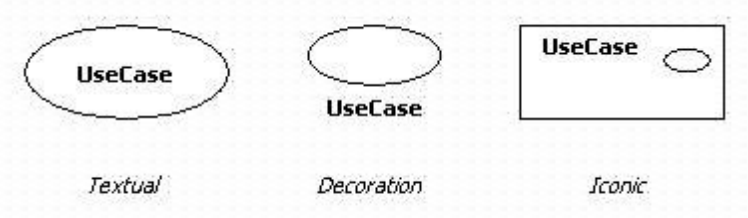
Semantics

The use case construct is used to define the behavior of a system or other semantic entity without revealing the entity's internal structure. Each use case specifies a sequence of actions, including variants, that the entity can perform, interacting with actors of the entity.

Procedure for creating UseCase

In order to create UseCase, click **[Toolbox]** -> **[UseCase]** button and click the position where to place UseCase on the **[main window]**.

UseCase is expressed in the forms of textual, decoration, iconic. To change UseCase's view style, select menu item under **[Format] -> [Stereotype Display]** or select  button's combo item.



Procedure for adding Extension

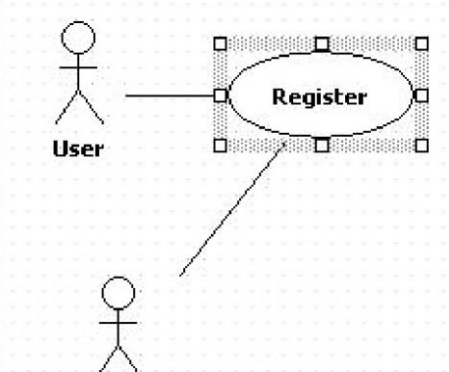
An extension point references one or a collection of locations in a use case where the use case may be extended.

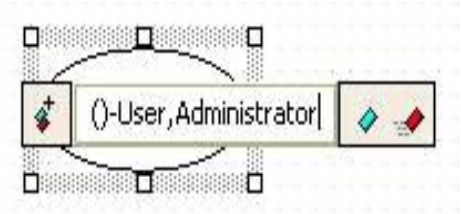
To edit ExtensionPoints of UseCase, click UseCase's **[Collection Editor...]** popup menu or click button of **[ExtensionPoints]** collection property.

Procedure for creating Actor from UseCase

In order to create multiple Actors related to UseCase at once, use shortcut creation syntax.

1. Double-click UseCase, or select UseCase and press **[Enter]** key. At quick dialog, enter Actor's name after "()-" string and separate Actor names by "," character.
2. And press **[Enter]** key. Several Actors associated with the UseCase are created and arranged vertically.





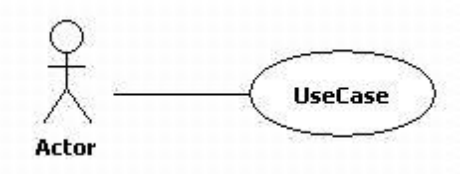
ASSOCIATION / DERECTED ASSOCIATION

Semantics

A association is an association among exactly two classifiers (including the possibility of an association from a classifier to itself).

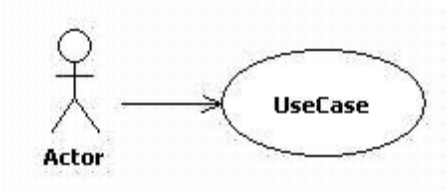
Procedure for creating association

In order to create association, click [Toolbox] -> [UseCase] -> [Association] button, drag from first element, and drop to second element in the [main window].

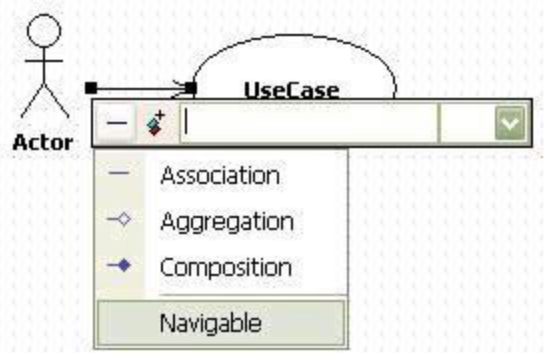


Procedure for creating directed association

The procedure is equal to the association's, drag and drop in the arrow direction.



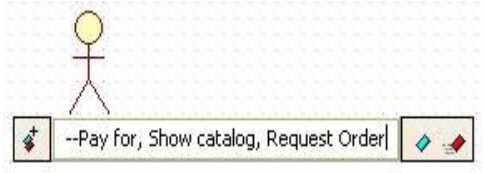
Or create association, click the actor-side association end. At the quick dialog, uncheck navigable and association becomes directed.

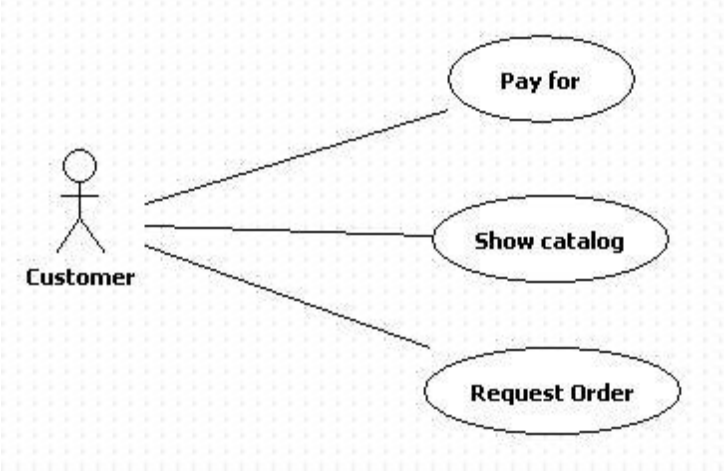


Procedure for creating element related to association/directed assocition

In order to create element associated with current element, use shortcut creation syntax.

1. Double-click element and enter element's names associated after "--" or "->" string at the quick dialog. Separate element names with "," character to relate multiple elements.
2. Press **[Enter]** key and several elements associated with selected element are created and arranged automatically.





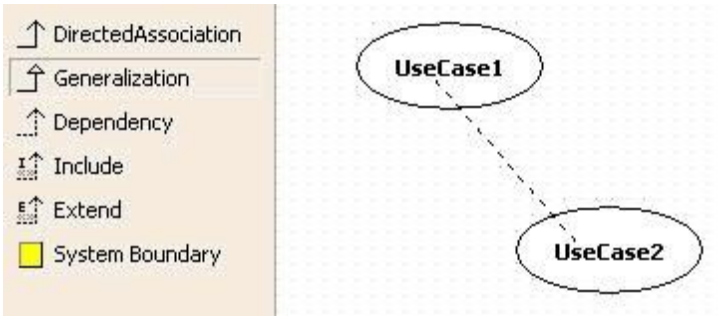
GENERALIZATION

Semantics

Generalization is the taxonomic relationship between a more general element (the parent) and a more specific element (the child) that is fully consistent with the first element and that adds additional information.

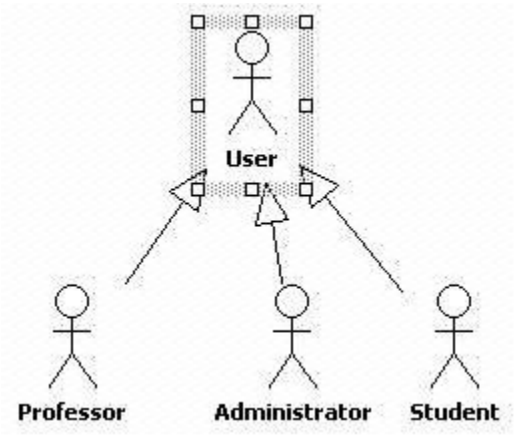
Procedure for creating generalization

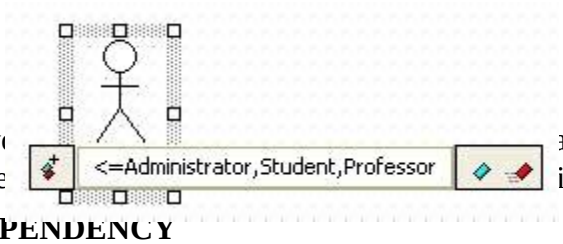
In order to make generalization, click **[Toolbox] -> [UseCase] ->[Generalization]** button, drag from child element and drop to parent element in the **[main window]**.



Procedure for creating multiple child actors inherited from actor To create multiple elements inherited from some element,

1. Enter with "<=" string as following at the quick dialog, and several elements inherited from selected element are created at once.
2. Child elements are generated below selected element and arranged automatically.



If y 

ıt once, ente ialog.

DEPENDENCY

Semantics

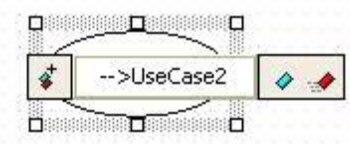
A *dependency* is a type of relationship that signifies that one element, or group of elements, acting as the client depends on another element or group of elements that act as a supplier. It is a weak relationship that denotes that if the supplier is changed the client may be affected. It is a unidirectional relationship.

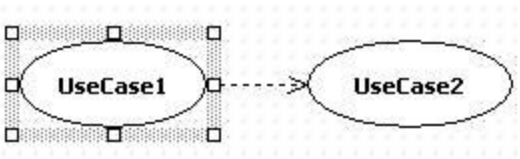
Procedure for creating dependency

In order to create dependency, click **[Toolbox] -> [UseCase] -> [Dependency]** button, drag element and drop to other element depended.

Procedure for creating other usecase depended by current usecase Enter with "-->" string at the quick dialog as following.

So dependency relationship is created between two elements.





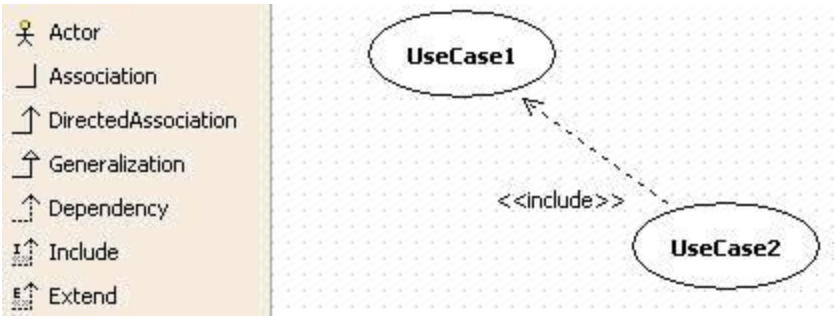
INCLUDE

Semantics

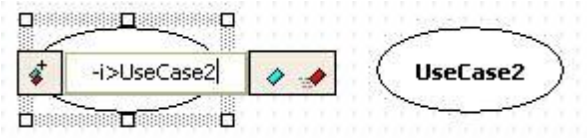
An include relationship defines that a use case contains the behavior defined in another use case.

Procedure for creating include

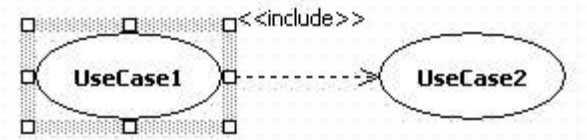
In order to create include relationship, click **[Toolbox] -> [UseCase] -> [Include]** button, drag from element including and drop to element included in the **[main window]**.



Procedure for creating other usecase included by current usecase Enter with "-i>" string at the quick dialog as following.

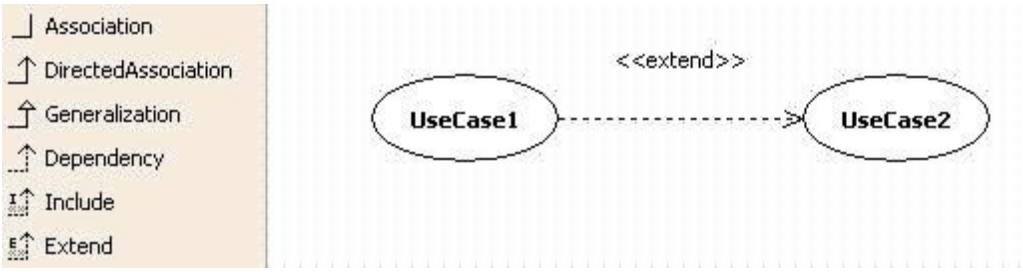


So include relationship is created between two elements.



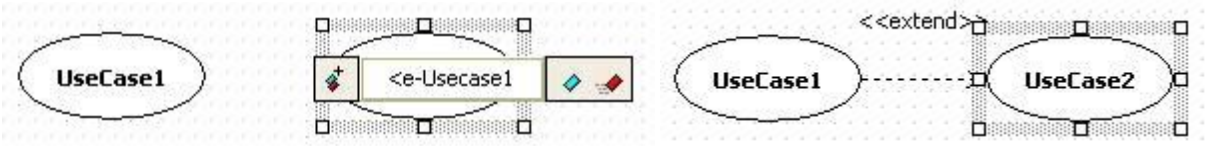
Procedure for creating extend

In order to create extend, click **[Toolbox]** -> **[UseCase]** -> **[Extend]** button, drag from element extending and drop to element extended in the **[main window]**.



Procedure for creating other usecase extending current usecase

Enter with "<e-" string at the quick dialog as following. So extend relationship is created between two elements.



Modeling with Class Diagram

The following elements are available in the class diagram. ➢
Subsystem

- Package
- Class
- Interface
- Enumeration
- Signal
- Exception
- Port
- Part
- Association
- DirectedAssociation
- Aggregation
- Composition
- Generalization
- Dependency
- Realization
- AssociationClass
- Connector
- Object
- Link

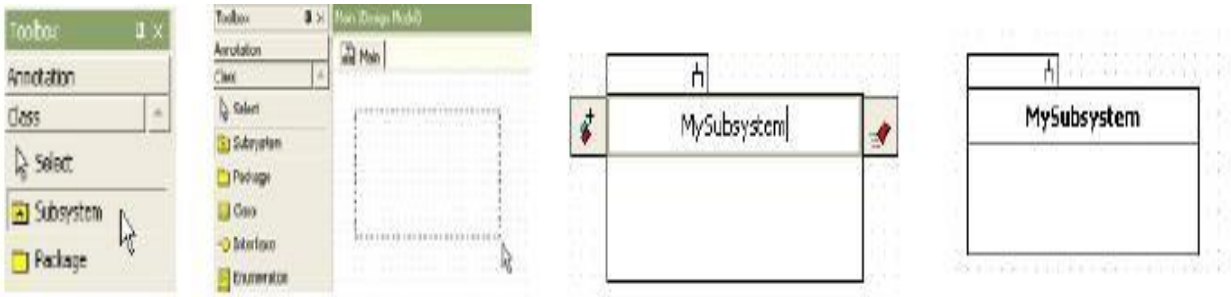
Semantics

Whereas a package is a generic mechanism for organizing model elements, a subsystem represents a behavioral unit in the physical system, and hence in the model.

Procedure for creating subsystem

In order to create subsystem,

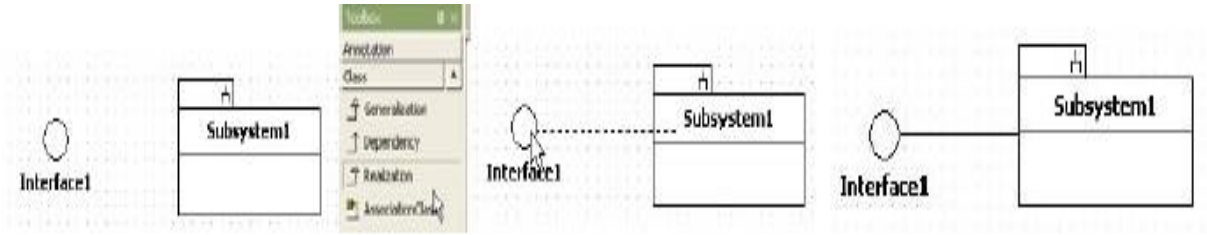
1. Click **[Toolbox]** -> **[Class]** -> **[Subsystem]** button.
2. And click at the location or boundary where subsystem will be placed in the **[main window]**.
3. Then a subsystem is created on the class diagram and subsystem quick dialog is opened. At the quick dialog, enter the subsystem name.
4. And press **[Enter]** key to have done this procedure.



Procedure for creating providing interface of subsystem.

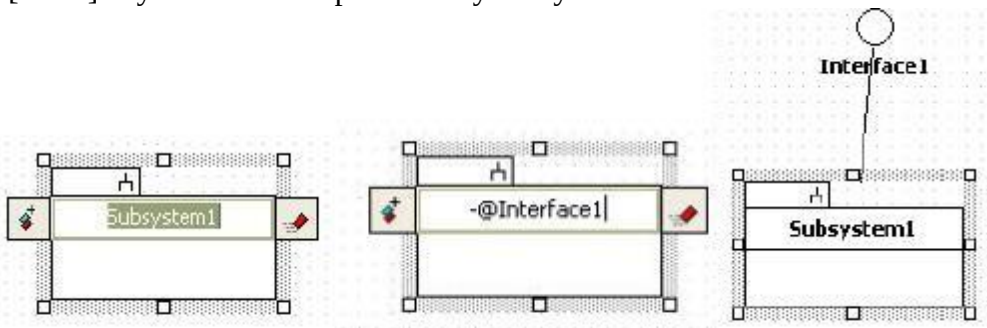
In order to providing inteface of subsystem,

1. Create interface and susbystem.
2. Click **[Toolbox] -> [Realization]** button.
3. Drag from subsystem and drop to interface.
4. Between interface and subsystem, providing interface relationship is created finally.



In order to create interface and realization at once,

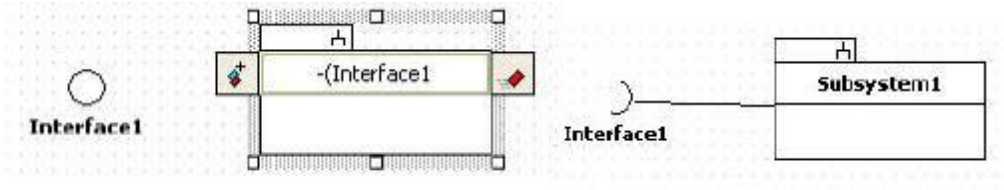
1. Double-click subsystem and subsystem quick dialog is opened.
2. Enter text in the quick dialog as following
3. Press [Enter] key and interface provided by subsystem is created . 50



Procedure for creating requiring interface

In order to create requiring interface, use shortcut creation syntax.

1. Double-click subsystem. At the quick dialog, enter text as follows.
2. Then subsystem connects to interface as requiring relationship.

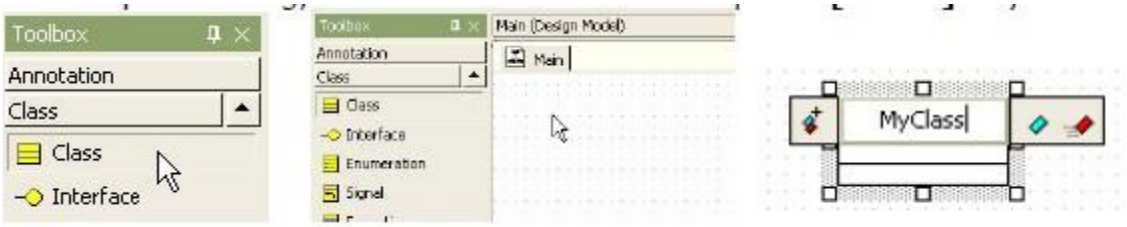


A class is the descriptor for a set of objects with similar structure, behavior, and relationships.

Procedure for creating class

In order to create class,

1. Click [Toolbox] -> [Class] -> [Class] button.
2. And click at the position where class will be placed in the [main window].
3. A new class is created on the diagram and class quick dialog is opened.
4. At the quick dialog, enter the class name and press [Enter] key.



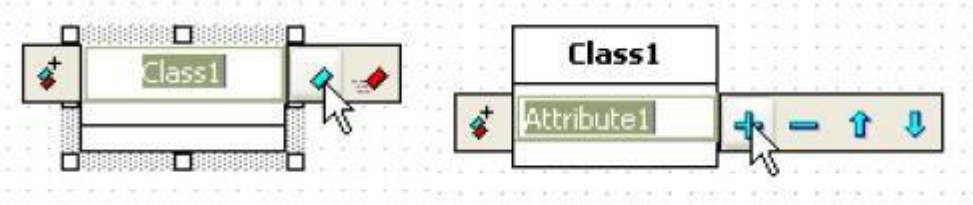
Procedure for adding attribute

There are three method to add attribute to class.

- using quick dialog
- using model in the [main window] or the [model explorer]
- using [collection editor] In the case of using quick dialog,

1. Double-click class.

2. Press [Add Attribute] button at the quick dialog, and you can add attribute.



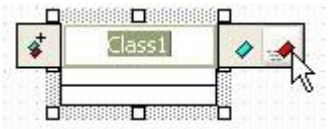
Procedure for adding operation

There are three method to add attribute to class.

• using quick dialog

• using model in the [main window] or the [model explorer]

• using [collection editor]
In the case of using quick dialog,
1. Double-click class and class quick dialog is shown.
2. Press [Add Operation] button at the quick dialog, and you can add operation.



Procedure for adding parameter to operation In order to add parameter to operation,

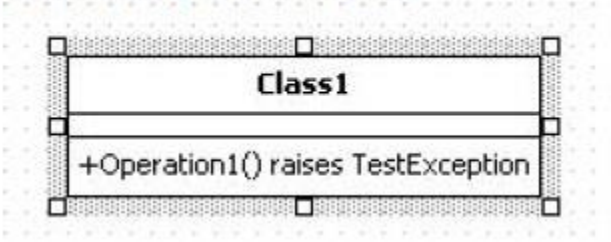
1. Select operation in the [model explorer], select [Add] -> [Parameter] popup menu, and new parameter will be added.
2. Or select operation in the [model explorer], select [Collection Editor...] popup menu.
3. Or click button in [Parameters] property on properties window.
4. At the [Parameters] tab of the [collection editor], you can add parameter by using button.

Procedure for adding exception to operation:

Before this procedure, there must exist a exception or more. To do this, see "Procedure for creating signal" or "Procedure for creating expception".

1. **Click ... button in [RaisedSignals] property on properties window.**
2. **At [Raised Signals] tab of the [collection editor], you can add exception to the operation by using button.**
3. **At [Select a Signal] dialog, select signal or exception raised by operation and click [OK] button.**

4. The result is as follows.



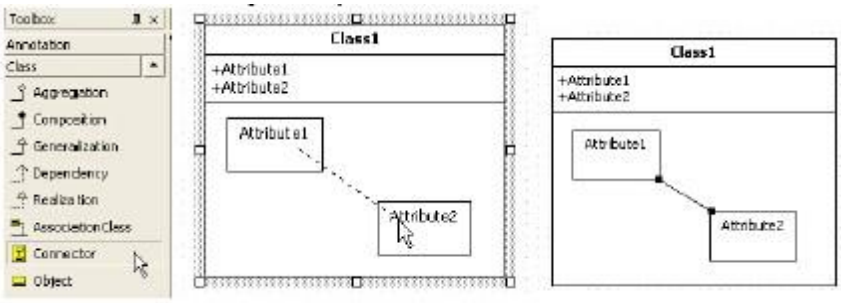
Procedure for creating connector

In order to create connector,

1. Click [Toolbox] -> [Class] -> [Connector] button.

2. Drag from one part and drop to the other part in the [main window].

3. Between two parts, new connector is created finally.

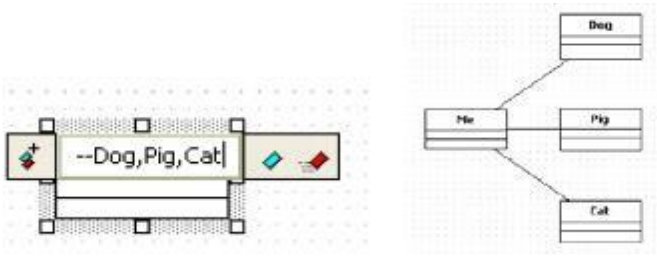


Procedure for creating multiple classes related to current class at once

If you want to create Dog, Pig, Cat classes related to Me class

1. Double-click Me class or press [Enter] key. At quick dialog, enter as following.

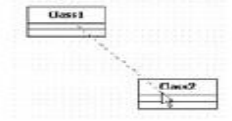
2. Then three classes with association are created as following.



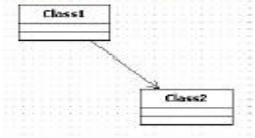
Procedure for creating directed association

Procedure for creating directed association is equal to association's.

1. **Click [Toolbox] -> [Class] -> [DirectedAssociation].**
2. **Drag and drop between two elements in arrow direction.**

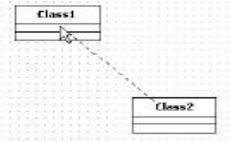
3. **The result is as follows.**



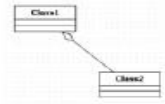
Procedure for creating aggreate

In order to create aggregation,

1. **Click [Toolbox] -> [Class] -> [Aggregation] button.**
2. **Drag from one associated and drop to another in the [main window].**

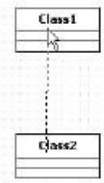
3. **The result is as follows.**

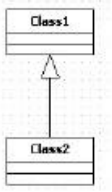


Procedure for creating generalization

In order to create generalization,

1. **Click [Toolbox] -> [Class] -> [Generalization] button.**
2. **Drag from child element and drop to parent element in the [main window].**
3. **Then a new generalization is created.**

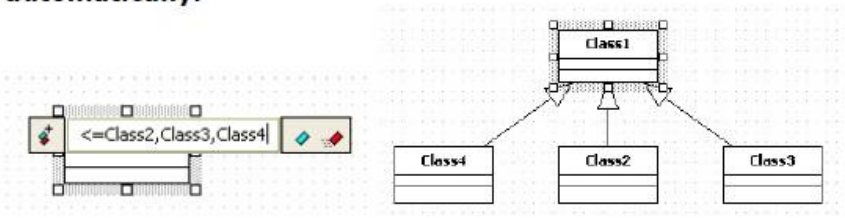




Procedure for creating multiple children classes at once.

In order to create multiple children classes inheriting selected class at once, use shortcut creation syntax.

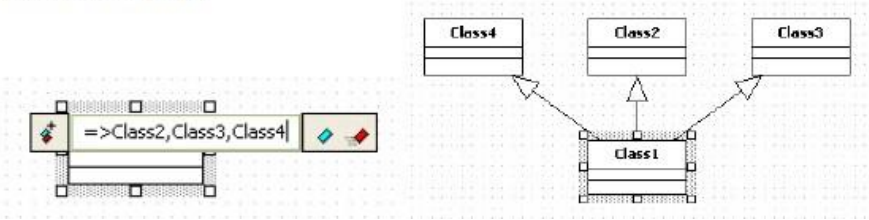
1. **Double-click to popup quick dialog. At the quick dialog, enter name of class inheriting selected class after "<=" string and separate names with ",".**
2. **The children classes are created below selected class and arranged automatically.**



Procedure for creating multiple parent classes at once

In order to create multiple parent classes of selected class at once, use shortcut creation syntax.

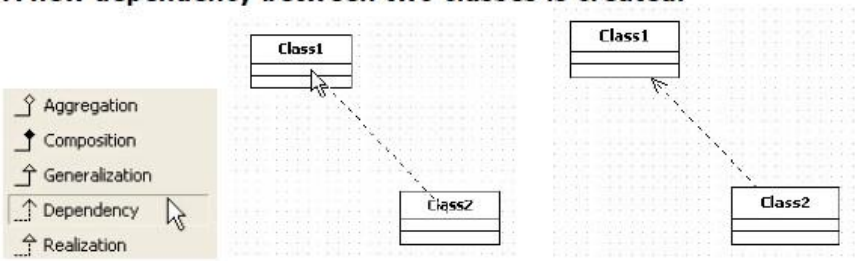
1. **Double-click to popup quick dialog. At the quick dialog, enter name of parent classes of selected class after "=>" string and separate names with ",".**
2. **The parent classes are created above selected class and arranged automatically.**



Procedure for creating dependency

In order to create dependency,

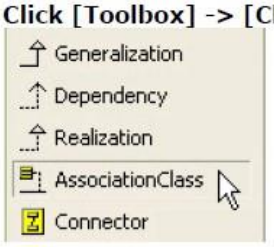
1. **Click [Toolbox] -> [Class] -> [Dependency] button.**
2. **Drag and drop between elements in the [main window] in depending direction.**
3. **A new dependency between two classes is created.**



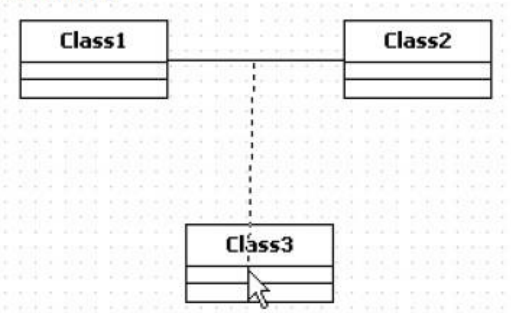
Procedure for creating association class

In order to create association class,

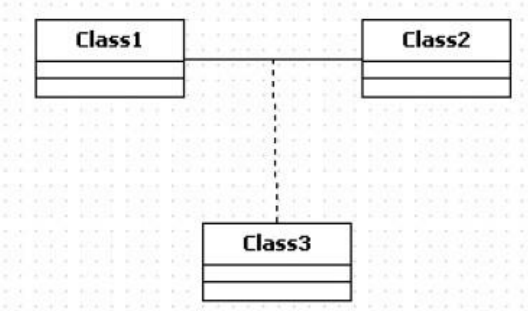
1. **Click [Toolbox] -> [Class] -> [AssociationClass] button.**



2. **Drag from association and drop to the class as association class in the [main window].**



3. **The result is as follows.**



OBJECT

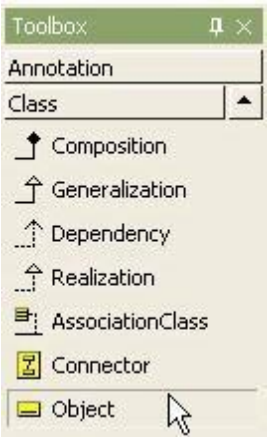
Semantics

An object represents a particular instance of a class. It has identity and attribute values. A similar notation also represents a role within a collaboration because roles have instance-like characteristics.

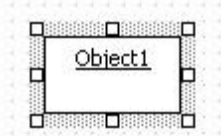
Procedure for creating object

In order to create object,

1. Click [Toolbox] -> [Class] -> [Object] button.



And click at the position where object will be placed in the [main window].



Procedure for adding AttributeLink to object

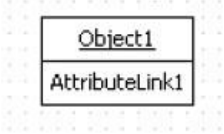
There are two way to add AttributeLink to Object.

• using object model in the [main window] or the [model explorer]

• using [collection editor]

In the case of using object model, select object in the [main window] or in the [model explorer], right-click the selected object, select [Add] -> [Attribute Link] popup menu, and you can add Attribute Link.

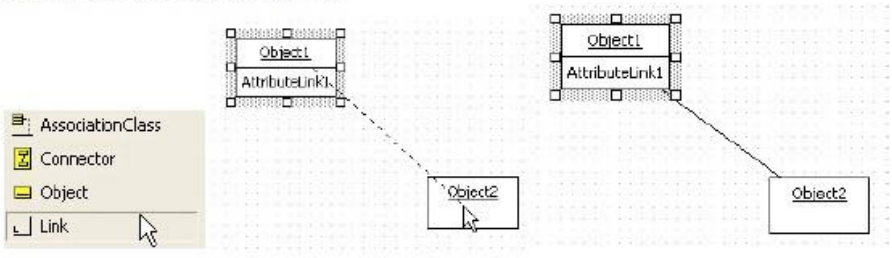
In the other case, select [Collection Editor...] popup menu of object or click ... button in slots property on properties window. At [Slots] tab of the [collection editor], you can add attribute link by using button.



Procedure for creating link

In order to create Link,

1. **Click [Toolbox] -> [Class] -> [Link] button.**
2. **Drag from one Object and drop to the other Object in the [main window].**
3. **The result is as follows.**



Modeling with Sequence Diagram

The following elements are available in a sequence diagram.

• Object

• Stimulus

• SelfStimulus

• Combined Fragment

• Interaction Operand

• FrameSubsytem

OBJECT

Procedure for creating object

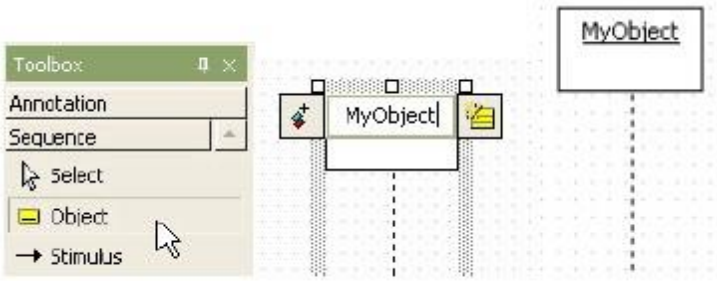
In order to create object,

1. Click [Toolbox] -> [Sequence] -> [Object] button.

2. And click at the position where object will be placed in the [main window].

3. Object quick dialog is shown. At the quick dialog, enter the object name.

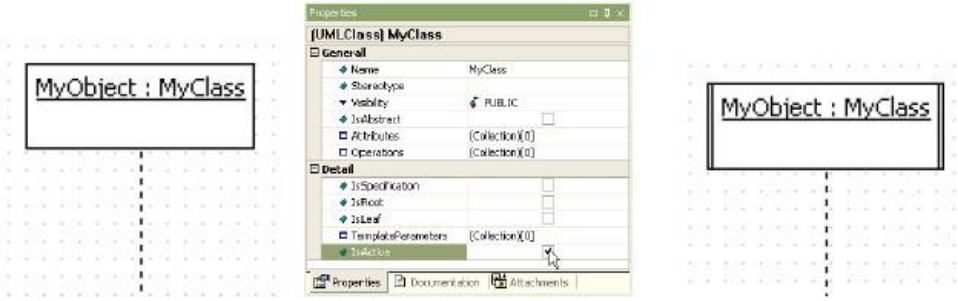
4. Press [Enter] key.



Procedure for setting active object

In order to set class to active object,

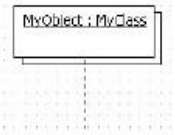
1. Set assigned class's **[IsActive]** property to true.
2. For MyObject, change MyClass's IsActive property.
3. If class property is not assigned, you can't change object to active object. The result is as follows.



Procedure for setting to multi object

In order to set object to multi object,

1. Set object's **[IsMultiInstance]** property to true.
2. Then the object is changed to multi object.

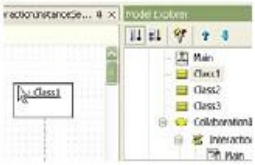



Procedure for creating object from class

In order to create object from class,

1. Select class in the **[model explorer]**.
2. Drag and drop it into **[main window]**.
3. Finally, a object is created on the diagram.

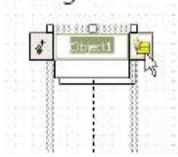


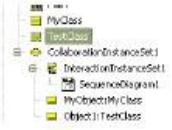


Procedure for creating class from object

If class is not assigned to object,

1. Double-click object to pop up quick dialog, click add class button
2. At the [Enter element name] dialog, enter the new class name.
3. And new class is created and assigned to object.
4. If you want existing class to be assigned to object, click ··· button in object's classifier property, and select class to be assigned to object at the **[Select a model element]** dialog.

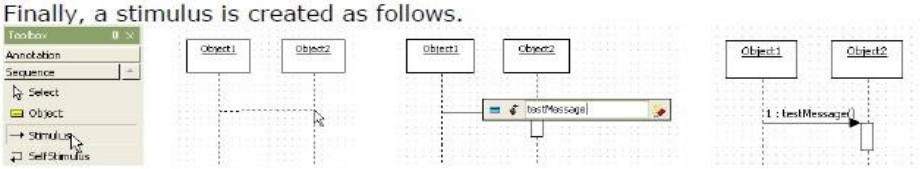


Procedure for creating stimulus

In order to create stimulus,

1. Click **[Toolbox] -> [Sequence] -> [Stimulus]** button.
2. Drag from one object, and drop to the other(object or lifeline) in the **[main window]** in outgoing direction.
3. Stimulus quick dialog is opened. Enter the stimulus name at the quick dialog and press **[Enter]** key.
4. Finally, a stimulus is created as follows.



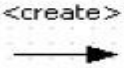
Procedure for using operation in class as stimulus

If classifier property of receiver(object) of stimulus is assigned and you want to assign operation to stimulus,

1. Double-click stimulus
2. Click button at the quick dialog.
3. Select operation on the **[Select an operation]** dialog, and click **[OK]** button.
4. New stimulus mapped to class's operation is added as follows.

Procedure for changing ActionKind of stimulus

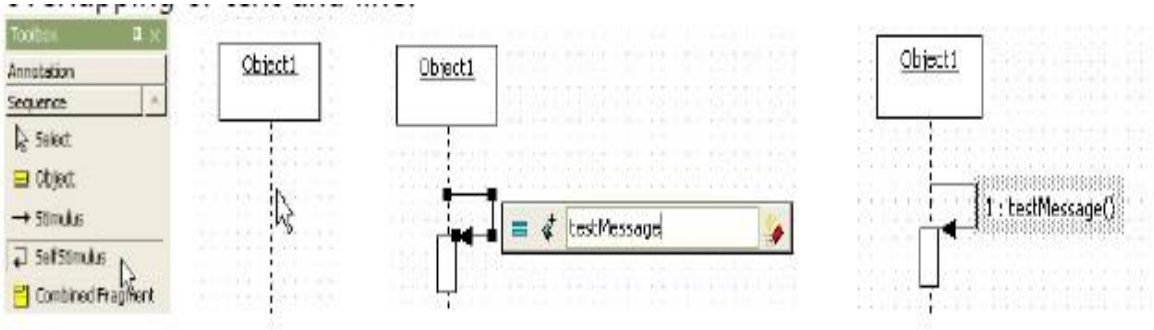
The [ActionKind] property of stimulus should be assigned to one of five sort as following. To change [ActionKind] property, select stimulus and select the [ActionKind] property on the properties window.

ActionKind	Shape
CALL	
SEND	
RETURN	
CREATE	<<create>> 
DESTROY	<<destroy>> 

Procedure for creating self-stimulus

In order to create self-stimulus,

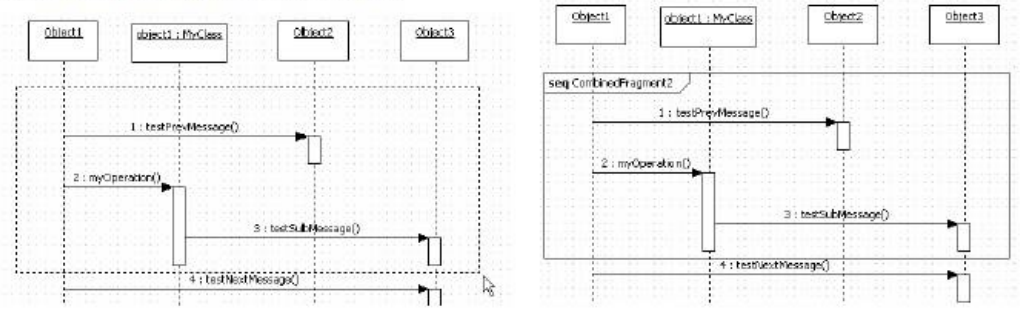
1. Click [Toolbox] -> [Sequence] -> [SelfStimulus] button.
2. And click the object(or lifeline) that self-stimulus will be placed in the [main window].
3. Object quick dialog is opened. At the quick dialog, enter the stimulus name and press [Enter] key.
4. The result of procedure is as follows. You may arrange stimulus position to reduce overlapping of text and line.



Procedure for creating combined fragment

In order to create Combined Fragment,

1. Click **[Toolbox] -> [Sequence] -> [Combined Fragment]** button.
2. And click at the position where Combined Fragment will be placed in the **[main window]**.
3. A combined fragment is created.

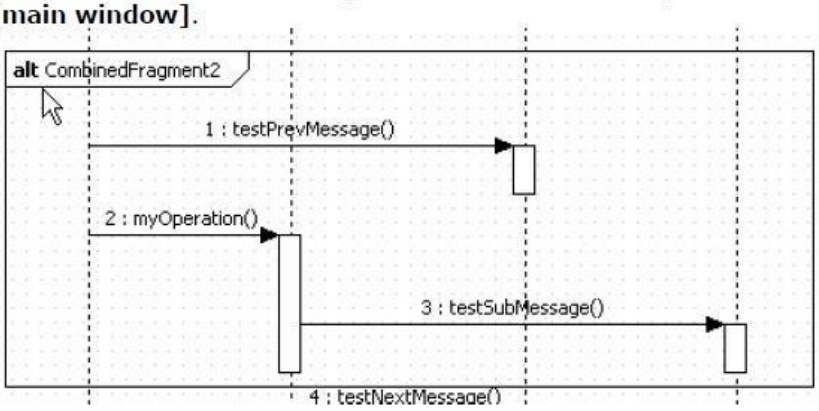


Procedure for creating interaction operand

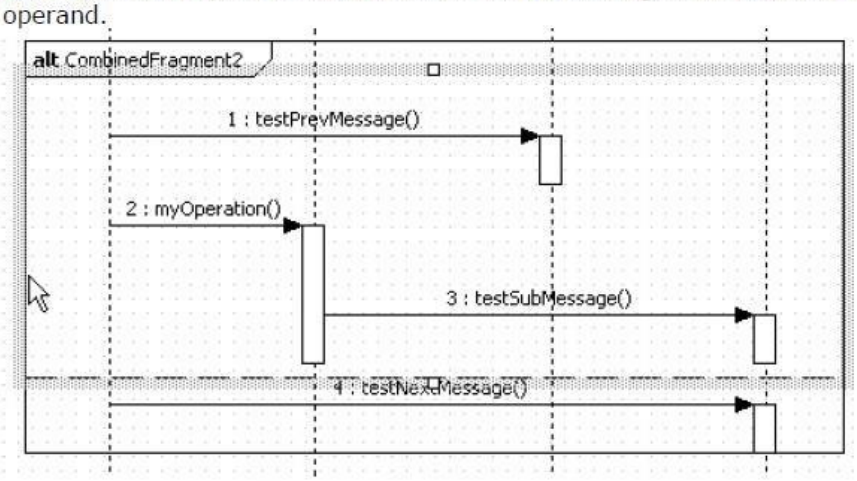
In order to create Interaction Operand,

1. Click **[Toolbox] -> [Sequence] -> [Interaction Operand]** button.

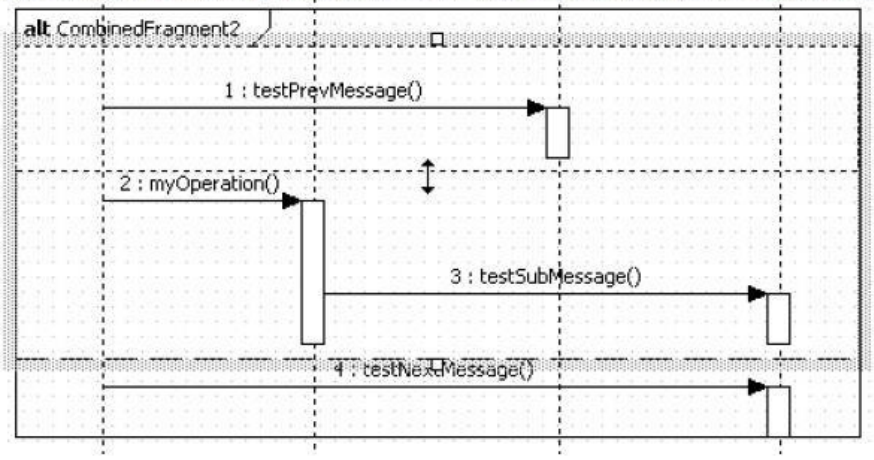
2. And click at the Combined Fragment where Interaction Operand will be placed in the **[main window]**.



3. New interaction operand is added to the combined fragment. Click the interaction operand.



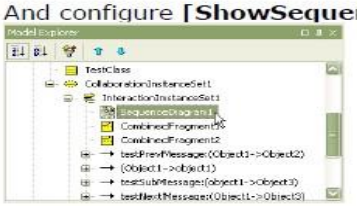
4. The selection points of interaction operand are shown, drag it to arrange its boundary.

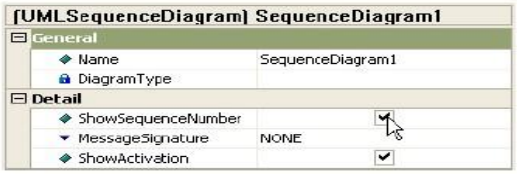


Procedure for showing sequence numbers in the diagram

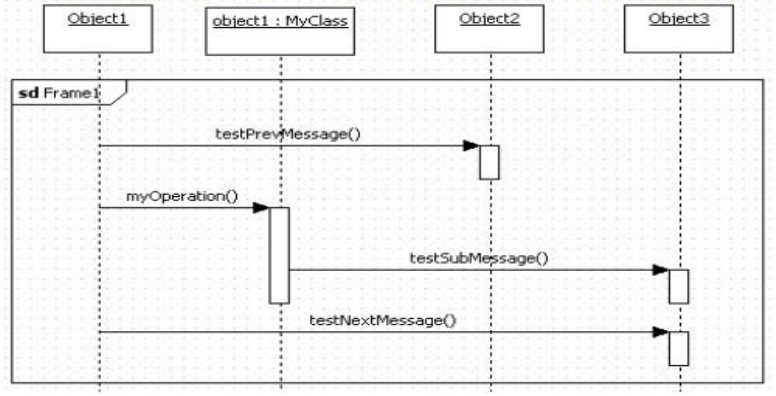
In order to show or hide stimulus sequence number,

1. Select the diagram in the **[model explorer]** or in the **[main window]**
2. And configure **[ShowSequenceNumber]** property of diagram to true or false.



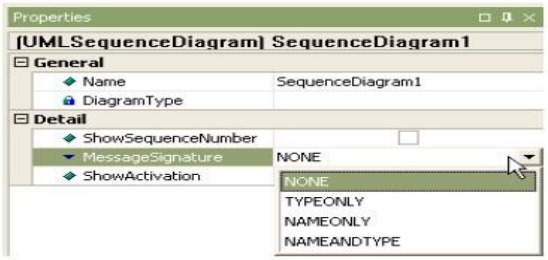


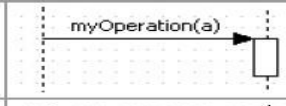
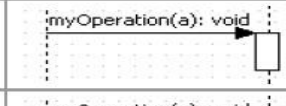
3. When **[ShowSequenceNumber]** is false, sequence diagram is shown as follows.



Procedure for changing signature style of message in the diagram

There are four message style. To change stimulus signature, select the diagram in the **[model explorer]** or in the **[main window]**, and configure **[MessageSignature]** property of diagram to one of the followings.



Style	Example
NONE	
NAMEONLY	
TYPEONLY	
NAMEANDTYPE	

Modeling with Collaboration Diagram

The following elements are available in a collaboration diagram.

• Object

• Link

• SelfLink

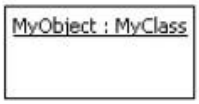
• Stimulus

• Frame

Procedure for setting active object

In order to set class to active object,

1. Set assigned class's **[IsActive]** property to true.
2. For MyObject, change MyClass's **[IsActive]** property.

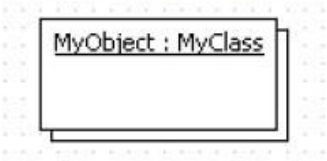


3. If class property is not assigned, you can't change object to active object.

Procedure for setting to multi object

In order to set object to multi object,

1. Set object's IsMultiInstance property to true.
2. Then the object is assigned as multi object.

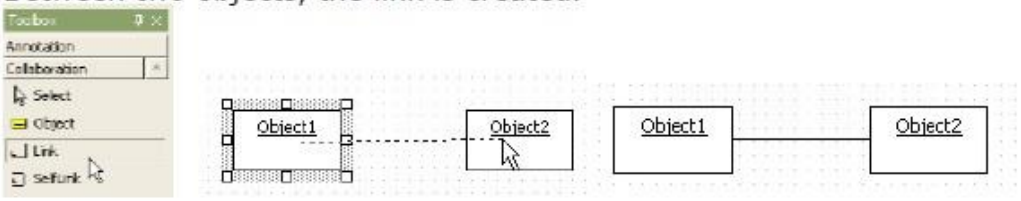


LINK

Procedure for creating link

In order to create Link,

1. Click **[Toolbox] -> [Collaboration] -> [Link]** button.
2. Drag from one Object and drop to the other Object in the **[main window]**.
3. Between two objects, the link is created.



Modeling with Activity Diagram

The following elements are available in a activity diagram.

- ActionState
- SubactivityState
- InitialState
- FinalState
- Synchronization
- Decision
- Flow Final
- Object Flow
- Signal Accept State
- Signal Send State
- Transition
- SelfTransition
- Swimlane

ACTIONSTATE

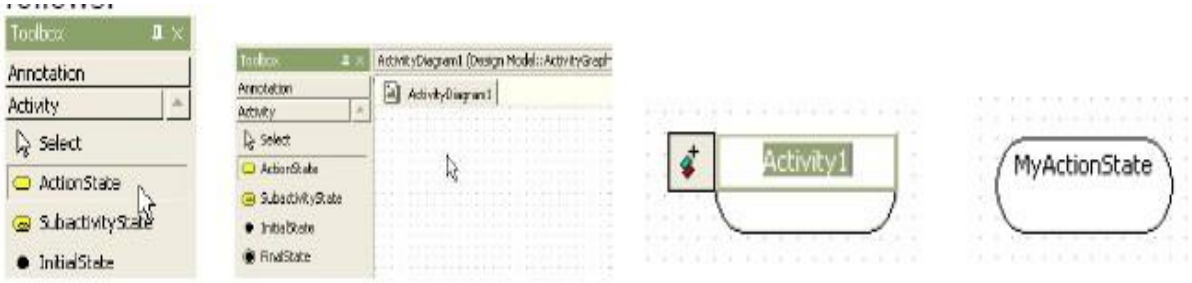
Semantics

An action state represents the execution of an atomic action, typically the invocation of an operation. An action state is a simple state with an entry action whose only exit transition is triggered by the implicit event of completing the execution of the entry action. The state therefore corresponds to the execution of the entry action itself and the outgoing transition is activated as soon as the action has completed its execution.

Procedure for creating action state

In order to create ActionState,

1. Click **[Toolbox] -> [Activity] -> [ActionState]** button.
2. And click at the position where ActionState will be placed in the **[main window]**.
3. A action state is created on the diagram and the quick dialog is shown.
4. Enter the action state name at the quick dialog and press **[Enter]** key. The result is as follows.



SUBACTIVITYSTATE

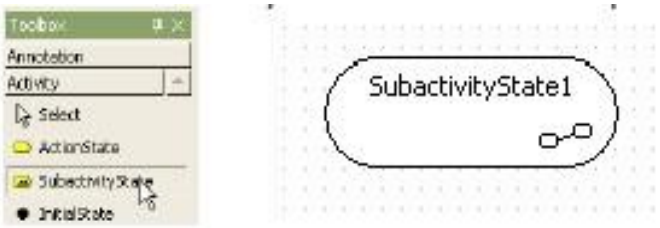
Semantics

A subactivity state represents the execution of a non-atomic sequence of steps that has some duration; that is, internally it consists of a set of actions and possibly waiting for events. That is, a subactivity state is a "hierarchical action," where an associated subactivity graph is executed.

Procedure for creating subactivity state

In order to create SubactivityState,

1. Click **[Toolbox] -> [Activity] -> [SubactivityState]** button.
2. And click at the position where SubactivityState will be placed in the **[main window]**. A subactivity state is created and the quick dialog is shown. At the quick dialog, enter the subactivity state name and press **[Enter]** key. The result is as follows.



INITIALSTATE

Procedure for creating initial state

In order to create InitialState,

1. Click **[Toolbox] -> [Activity] -> [InitialState]** button.
2. And click at the position where InitialState will be placed in the **[main window]**. Then a initial state is created.


FINALSTATE

Procedure for creating final state

In order to create FinalState,

1. Click **[Toolbox] -> [Activity] -> [FinalState]** button.
2. And click at the position where FinalState will be placed in the **[main window]**.


DECISION

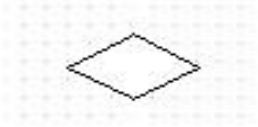
Semantics

A state diagram (and by derivation an activity diagram) expresses a decision when guard conditions are used to indicate different possible transitions that depend on Boolean conditions of the owning object.

Procedure for creating decision

In order to create Decision,

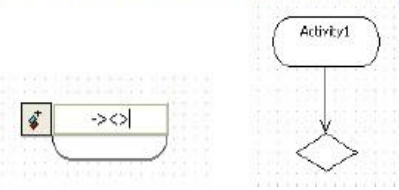
1. Click **[Toolbox] -> [Activity] -> [Decision]** button.
2. And click at the position where Decision will be placed in the **[main window]**. The decision is created on the diagram.

Procedure for creating decision from state

In order to create decision with incoming transition from selected object, use shortcut creation syntax.

1. Double-click state. At the quick dialog, enter "-><>"("<-<>" for incoming from decision) string.
2. Press **[Enter]** key and decision with outgoing transition from selected state is created.

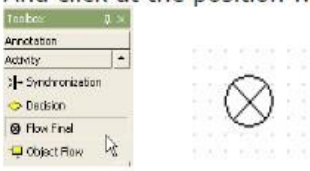


FLOW FINAL

Procedure for creating flow final

In order to create Flow Final,

1. Click **[Toolbox] -> [Activity] -> [Flow Final]** button.
2. And click at the position where Flow Final will be placed in the **[main window]**.



OBJECT FLOW

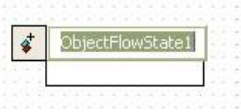
Semantics

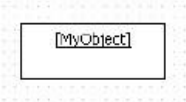
An object flow is one of two types of activity edges, which are directed connection (flows) between activity nodes, the other being a control flow. As soon as the activity node at the source (tail) end of the flow is finished it presents tokens to the object flow at the target (arrowhead) end of the flow. An object flow can only carry object (data) tokens; it cannot carry control tokens. There are rules that specify whether tokens can flow along the object flow and these are determined by the type of activity node at the source and target of the flow. In the case of complete activities an object flow may define a weight, which specifies the minimum number of tokens that must flow along the object flow as a group.

Procedure for creating object flow

In order to create Object Flow,

1. Click **[Toolbox] -> [Activity] -> [Object Flow]** button.
2. And click at the position where Object Flow will be placed in the **[main window]**. Then the quick dialog of object flow state is shown as follows.
3. At the quick dialog, enter the object flow state name and press **[Enter]** key.





SYNCHRONIZATION

Procedure for creating synchronization bar

In order to create Synchronization,

1. Click **[Toolbox] -> [Activity] -> [Synchronization]** button.
2. And click at the position where Synchronization will be placed in the **[main window]** and drag as size as you want.
3. The following figure shows the result of this procedure.

SIGNAL ACCEPT STATE

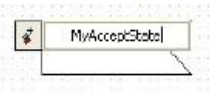
Semantics

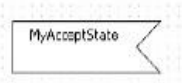
The signal accept may be shown as a concave pentagon that looks like a rectangle with a triangular notch in its side (either side). The signature of the signal is shown inside the symbol. An unlabeled transition arrow is drawn from the previous action state to the pentagon and another unlabeled transition arrow is drawn from the pentagon to the next action state. A dashed arrow may be drawn from an object symbol to the notch on the pentagon to show the sender of the signal; this is optional.

Procedure for creating signal accept state

In order to create Signal Accept State,

1. Click **[Toolbox] -> [Activity] -> [Signal Accept State]** button.
2. And click at the position where Signal Accept State will be placed in the **[main window]**.
3. At the quick dialog, enter signal accept state name and press **[Enter]** key.





LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

FLIPKART

-If it's trendy, it's on Flipkart. Be trendy, always.

PROBLEM STATEMENT:

In a fast economic growing country like India people tend towards convenient shopping portals such as eCommerce, by ecommerce companies the shopping has been organized in a structured manner. To solve the problems of quality of products, maintaining data servers, compensating sellers, to achieve best customer service, to manage shipping of goods to implement reliable payment services, Flipkart is set out to provide the best online shopping experience all over its customers across Indian region.

ABSTRACT

The Indian E-commerce market has undergone exhaustive changes in the recent times. E-commerce has reached the doorstep of a common individual in India. It has broken the technological and geographical barriers over the years and has got huge amount of success. The E-Retailing form of market was something unheard of to the Indian consumer in 2007.Many Indians today are embracing e-retailing with enthusiasm. Popular portals such as Flipkart are spearheading the conversion of offline shoppers into online bargain hunters.

Flipkart has made a name for itself in terms of market share and goodwill in the online market and it is posing a big threat to the retailers by its marketing strategies. Flipkart.com will continue to enhance and broaden its brand, customer base and electronic commerce expertise with the goal of creating customers preferred online shopping destination around India. This case study aims to study functioning, market strategy, SWOT analysis of Flipkart.

Keywords: E-commerce, E-retailing, E-business.

1.INRODUCTION

E-Commerce, also known as electronic commerce or internet commerce, is an activity of buying and selling goods or services over the internet or open networks. So, any kind of transaction (whether money, funds, or data) is considered as E-commerce. So, E-commerce can be defined in many ways, some define E-Commerce as buying and selling goods and services over the Internet, others define E-Commerce as retail sales to consumers for which the transaction takes place on open networks. The buying and selling of products, services, and digital products through the Internet all fall under the umbrella of e-commerce.

E-commerce is powered by the internet, where customers can access an online store to browse through, and place orders for products or services via their own devices.

As the order is placed, the customer's web browser will communicate back and forth with the server hosting the online store website. Data pertaining to the order will then be relayed to a central computer known as the order manager-- then forwarded to databases that manage inventory levels, a merchant system that manages payment information (using applications such as PayPal), and a bank computer -- before circling back to the order manager. This is to make sure that store inventory and customer funds are sufficient for the order to be processed. After the order is validated, the order manager will notify the store's web server, which will then display a message notifying the customer that their order has been successfully processed. The order manager will then send order data to the warehouse or fulfillment department, in order for the product or service to be successfully dispatched to the customer. At this point tangible and/or digital products may be shipped to a customer, or access to a service may be granted.

Platforms that host e-commerce transactions may include online marketplaces that sellers simply sign up for, such as Amazon.com; software as a service Tools that allow customers to 'rent' online store infrastructures; or open-source tools for companies to use in-house development to manage.

E-commerce is the application of information technology and communication technology to three basic activities related to commercial business; the three basic activities are as follows:

- Production and support- which includes assisting production, distribution, and maintenance of goods and services.

- Transaction preparation- which includes getting product information into the market-place and bringing buyers and sellers into contract with each other; and
- Transaction completion- which includes concluding transactions, transferring payments, and securing financial services.

E-business applications turn into e-commerce precisely, when an exchange of value occurs. Digitally enabled transactions include all transactions mediated by digital technology and platform; that is, transactions that occur over the Internet and the web.

Over the last decade the advent of e-commerce has actually transformed the manner in which people used internet. People now are not only just using internet for gathering information, leisure or socializing online but also at the same time they are seeking measures to conduct business.

E-commerce has grown rapidly and changed significantly over the years, as such there is plenty for buyers, investors and entrepreneurs to be mindful of. Understanding different fulfilment models, as well as the pros and cons of different platforms, is critical when evaluating potential e-commerce businesses for sale.

1.1SCOPE

E commerce Companies involved in electronic commerce as either buyers or sellers rely on Internet-based technologies and e-commerce applications and services to accomplish marketing, discovery, transaction processing and product and customer service processes.

1.2 OBJECTIVES

1.Development of Business-Relationship

2.Better-Customer Service

3.To Enhance the Efficiency of Services

4.Getting more Customers

5.Realistic and clear

6.Specific and sharp

7.Reflects the company’s offerings

8.Timely

9.Measurable

10.Attainable

1. Development of Business-Relationship:

The business development can be done through the e-commerce being the primary and the basic object. As their direct contact in between the company and the consumer, their business relationship will be enhanced. Hence the area of the market can be increased.

2. Better-Customer Service:

As it is done round the clock, the customer will always have online help regarding the products. As all the information is furnished to the customer, it becomes easy to him to choose the best product among all other alternatives. As even the service can also be done through the net immediately, the customer service will be ballooned. By highlighting the customer service, the companies are trying to subjugate a lion-share in the market.

3.To Enhance the Efficiency of Services:

By opting for the online E-commerce platform, you can boost up your efficiency. Opting for Ecommerce not only increases your sales but also helps as a cost-effective method. With Ecommerce, you can reduce your managing and warehousing cost. It eventually helps you save more funds at your disposal. You can also reduce delivery time with E-commerce and make your customers happy.

4. Getting more Customers:

In these days it becomes the mandate of the companies to double its customers, and this can be done by rendering the value-add service and maintaining the quality. Hence, it is also one of the primary objectives of the companies which supply impetus for the robust growth in sales and overall profit.

5. Realistic and clear:

This means that Flipkart com has used simple, string, and easily understood words and phrases in the drafting of its mission statement. Clarity is important so that the mission statement is understood by all relevant stakeholders of Flipkart com Company. Flipkart com's mission statement is also realistic, which makes it able to achieve various set goals and targets.

6.Specific and sharp:

It is easy to understand and delivers what the audience must know about Flipkart com's offerings and operations. It is important to keep the mission's statement short, sharp and precise to be able to successfully communicate the company's standing to stakeholders, instead of dragging it on into long pages with repetition and non-important aspects.

7. Reflects the company's offerings:

company should be based on what the company has to offer in terms of products and services. This means that the mission statement for Flipkart com highlights its offerings, but ensures that

this offering is in line with the values that the company stands for; therefore, identifies the ethical grounds through which the company systematically works to deliver its offering.

8.Timely:

Objectives at Flipkart com are also time-bound in that they have a specified start and finish date. The timeliness of the objective helps Flipkart com maintain a sense of urgency in employees, and keep them motivated towards achieving the objective.

9. Measurable:

Objectives at Flipkart com are also measurable. This means that all objectives can be tracked for progress. This is important for Flipkart com as it helps in meeting deadlines. The element of measurability is added in objectives by adding quantifiable criteria for determining progress and objective achievement.

10. Attainable:

The goal should be attainable that even in stretching the abilities of the employees and challenging them, it should remain possible to achieve. The objectives at Flipkart com are attainable in that they push the employees out of their comfort zones but remain possible to achieve.

2.LITERATURE REVIEW

2.1 HISTORY OF FLIPKART

It was founded in 2007 by Sachin Bansal and binny Bansal. Both alumni of the Indian institute of technology Delhi. They worked for amazon.com and left to create their new company incorporated in October 2007, as flip kart online service pvtltd. The first product they sold was the book "leaving Microsoft change the world to customer from Hyderabad". Flipkart now employees more than 33,000 people. Flipkart allows payment methods such as cash on delivery, credit or debit card transaction, net banking, e-gift voucher and card swipe on delivery. After failure of its 2014 big billion sale, it recently completed the second edition of big billion sale hold between October 13 and 17 where it is reported that they saw a business turnover of 300million in gross merchandise volume. It is registered in Singapore but has it headquartered in Bangalore, Karnataka, India. It launched its own product range under the name "DIGFLIP" with products including tablets, USB and laptop bags. In may 2014, Flipkart received $210 million from DST global, in July 2014 it raised $1 billion led by existing investors tiger global and south African's media group Naspers and in may 2015 it raised $550 million from some of its existing investors. Flipkart last fund-raising round in may 2015 has 18 pegged its valuation at $15 billion in february2016, merger Staley, marked down its investors value to $11 billion.

Sherah Kurnia (2003): According to his research paper about the acceptance of online grocery shopping the attitude of the potential consumer in terms of usefulness, continence and risk as well as the existence towards the social influence and visibility of technology influence the perception of customer towards online grocery shopping.

Akansha Pahwa (2015): According to her article, in Inc42.magazine Flipkart is little slow in arriving into the sector of online grocery as there are already many rivals which exits in the market. As already Ecommerce website like Paytm, ola were indulged in grocery sectors. The new players in this market are big basket, Peper Tap, Zop Now which uses different model and technology to mark its presence in th**e** industry.

2.2 The Journey of Flipkart

It's noteworthy to remember that Binny Bansal got rejected twice by Google. He worked for Amazon.com for nine months. Binny and Sachin understood that India's e-

commerce sector was little at first, so they decided to launch a "Comparison Search Engine." However, in 2007, they both quit Amazon.com and founded Flipkart, an e-commerce startup.

Customers appreciated Flipkart's assurance and trust element. The company expanded its rapid service and delivery became the company's hallmark as word spread in a positive tone. The business grew, and the e-commerce site established itself as India's leading platform.

Forbes India Rich List established its first recognition in 2015. Binny and Sachin Bansal were named India's 86th wealthiest persons. Their combined net worth was estimated to be $1.3 billion. Despite the presence of Amazon.com and other e-commerce platforms, their vision led them to higher goals, as the company continued to grow its business.

2.2.1 Awards and recognition

- The Economic Times named him Entrepreneur of the Year for 2012–2013.
- Founders debuted at #86 on Forbes' India Rich List, with a combined net worth of $1.3 billion.
- Founders were selected to Time magazine's annual list of the worlds' 100 most influential people in April 2016

2.2.2 Walmart Acquires Flipkart

In May 2018, Walmart, the American retail conglomerate, purchased a 77 percent stake in Flipkart. Binny Bansal, Tencent, Microsoft, and Tiger Global will own the remaining 23%. With a 77 percent ownership worth $16 billion, the young entrepreneurs have become one of India's most successful entrepreneurs.

Walmart has stated that it will finance the round with a mix of newly issued debt and cash. A total of $2 billion in new equity funding has been invested in the company. They also stated that both companies' branding and operating structures would be maintained.

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

eBay Inc Is an American multinational e-commerce corporation based in Sanjose, California, that facilitates consumer-to-consumer and business-to-consumer sales through its website. eBay was founded by Pierre Omidyar in 1995, and became a notable success story of the dot-com bubble.

3.1.1 Drawbacks:

- Anyone can sell on eBay, which means buyers can't be sure what type of customer service they'll receive. The seller might be slow to respond to questions.
- Uncertain cost and fees on delivery.
- From the seller's end, dealing with customers is sometimes a disadvantage of eBay. Some customers might complain about the smallest thing or be very demanding.
- It's very prone to fraud and scams, sellers intentionally mispresent the item of sale.

3.2 PROPOSED SYSTEM

Flipkart (Company) was founded in 2007 by Sachin Bansal and Binny Bansal, both alumni of the Indian Institute of Technology Delhi. They worked for Amazon.com, and left to create their new company incorporated in October 2007.

3.2.1 Advantages of Flipkart over Ebay:

- In Flipkart, the sellers are certified therefore the products are of estimated quality and they will get a productive response from the customer care.
- Fees and cost in Flipkart are determined at the time of ordering and its minimal.
- Flipkart assures its customers maximum satisfaction.
- The customer can return or replace the order based on his necessity.

3.3 TECHNICAL SPECIFICATION

Since it is built mainly on PHP, Java; Flipkart site enables a contented purchasing experience as in a real store. It uses MySQL for data storage and all of its software functions on Linux.

3.3.1 SOFTWARE REQUIREMENTS:

OPERATING SYSTEM:

WINDOWS

WINDOWS 7 or later.

ANDROID

Android 5.0 and above.

IOS

Requires iOS 11.0 or later.

3.3.2 HARDWARE REQUIREMENTS:

IOS

• 0.5 GB RAM and up to 6 GB memory and up

ANDROID

• 512 MB RAM and up to 850 MB memory and up

• 1Ghz processor

WINDOWS

• Any Operating System that supports web communication

4.SYSTEM DESIGN

4.1 UML DIAGRAMS:

Class diagram:

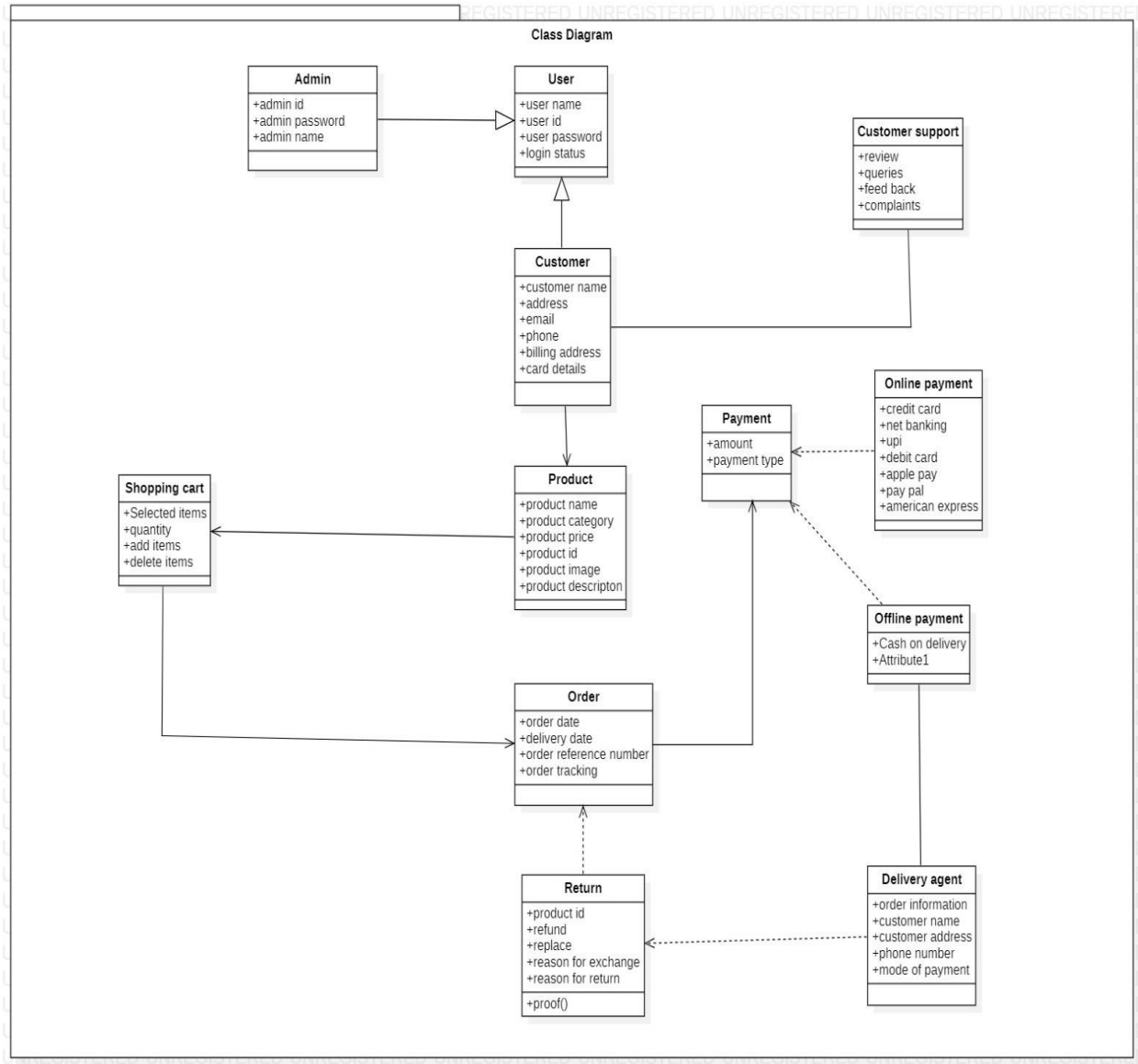


Figure 4.1.1: Class Diagram for Flipkart

This class his diagram of Flipkart represents the different modules, the functionality provided by them, and the relationships between them.

The above class describes the structure of Flipkart consists of classes, their attributes, operations and relationship among them. The main classes of Flipkart system are Customer, Cart, Product, Order, Payment, Customer care.

- Customer Class: Manage all the operations of customers.
- Product Class: Manage all the operations of attributes of product.
- Order Class: Manage all the orders of customer.
- Payment Class: Manage all the operations of payment and its types.
- Customer care Class: Handles the queries of customer.

USE CASE DIAGRAMS

Use case diagrams in unified modelling language helps us to identify and summaries the user's interaction with the system, it mainly contains actors and operations. In this Flipkart use case diagram show the scenarios of different actors interacting with the Flipkart system. Mainly three use case scenarios i.e,

USE CASE DIAGRAM 1:

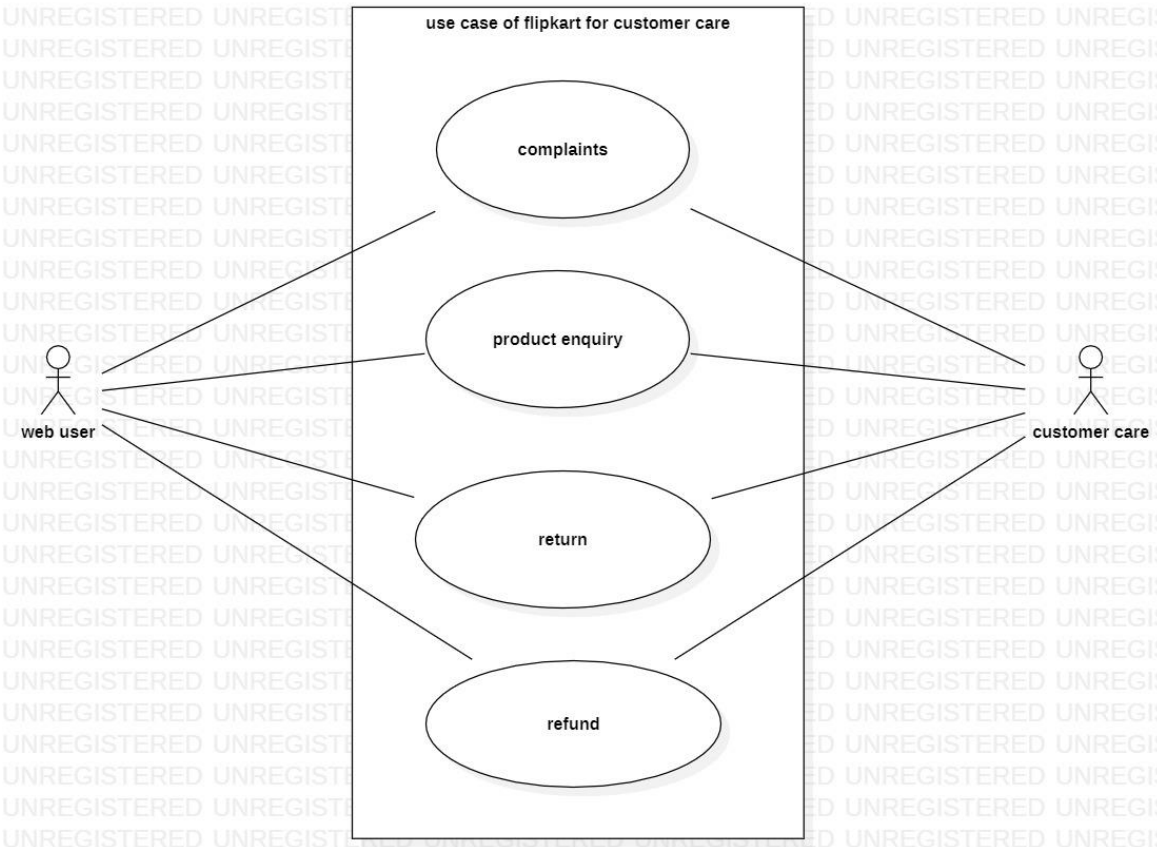


Figure 4.1.2 use case diagram for customer care

In this scenario the web user is using Flipkart customer care services such as the complaints regarding the products and orders, the required product enquiries, the time framed return and refund policy.

USE CASE DIAGRAM 2:

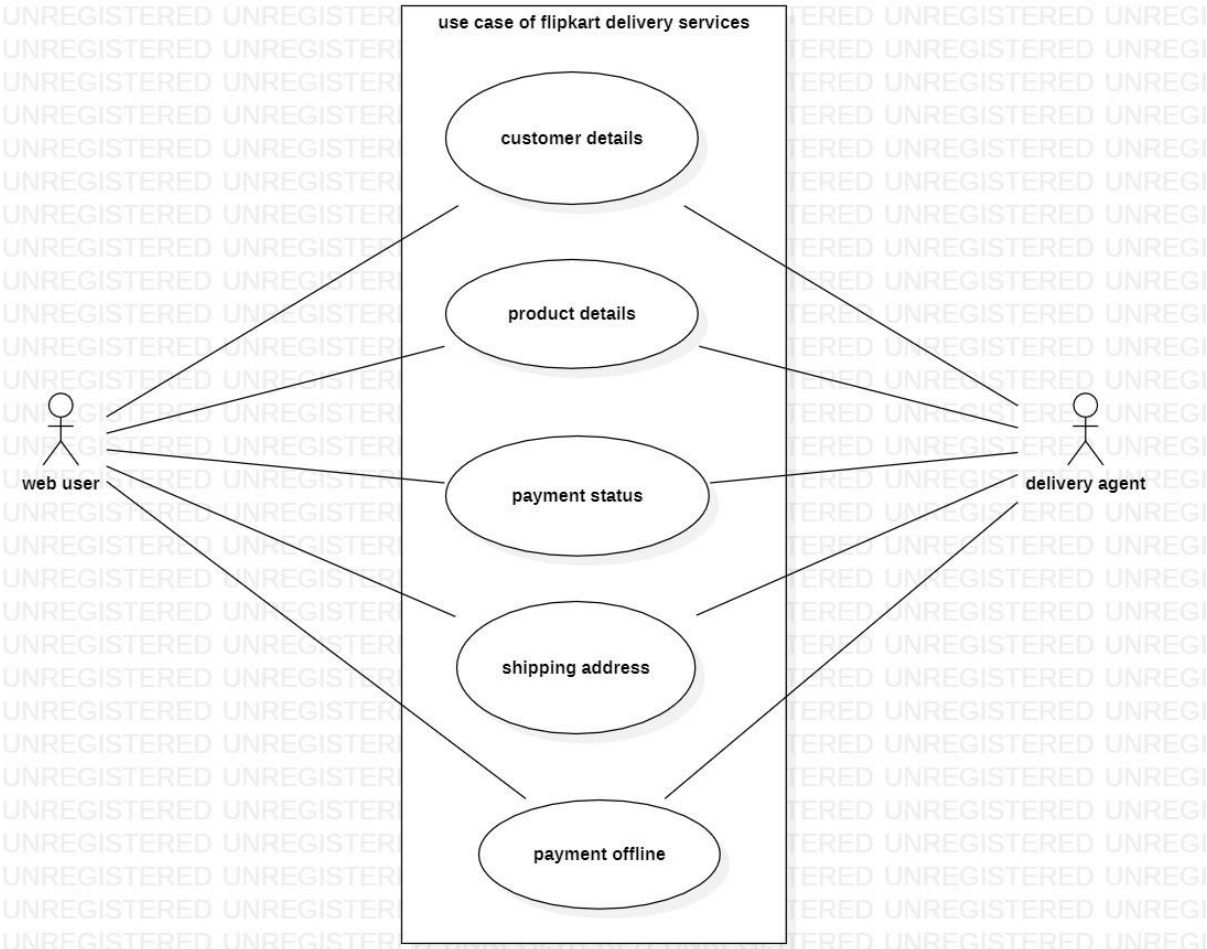


Figure 4.1.3: use case diagram for delivery

In this scenario the delivery agent is using the customer details, the product details, the payment status of the customer, the shipping address and also accepting the payment if it is done in offline mode.

USE CASE DIAGRAM 3:

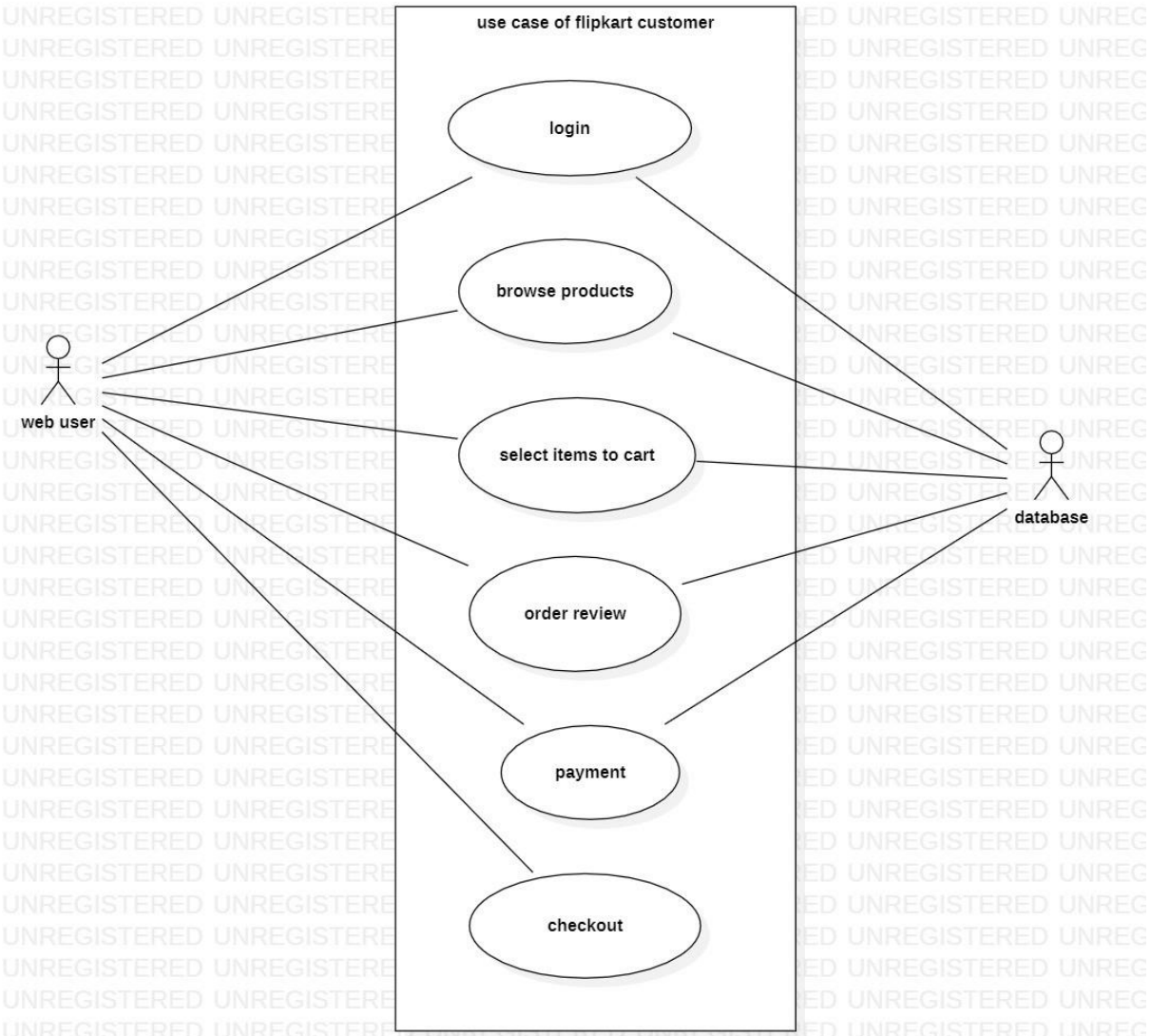


Figure 4.1.4: Use case diagram for Customer

In this scenario the customer or the web user uses flipkart portal for accessing the products, adding items to cart, verifying the order, and make payment for the suitable products required by the customer.

ACTIVITY DIAGRAM:

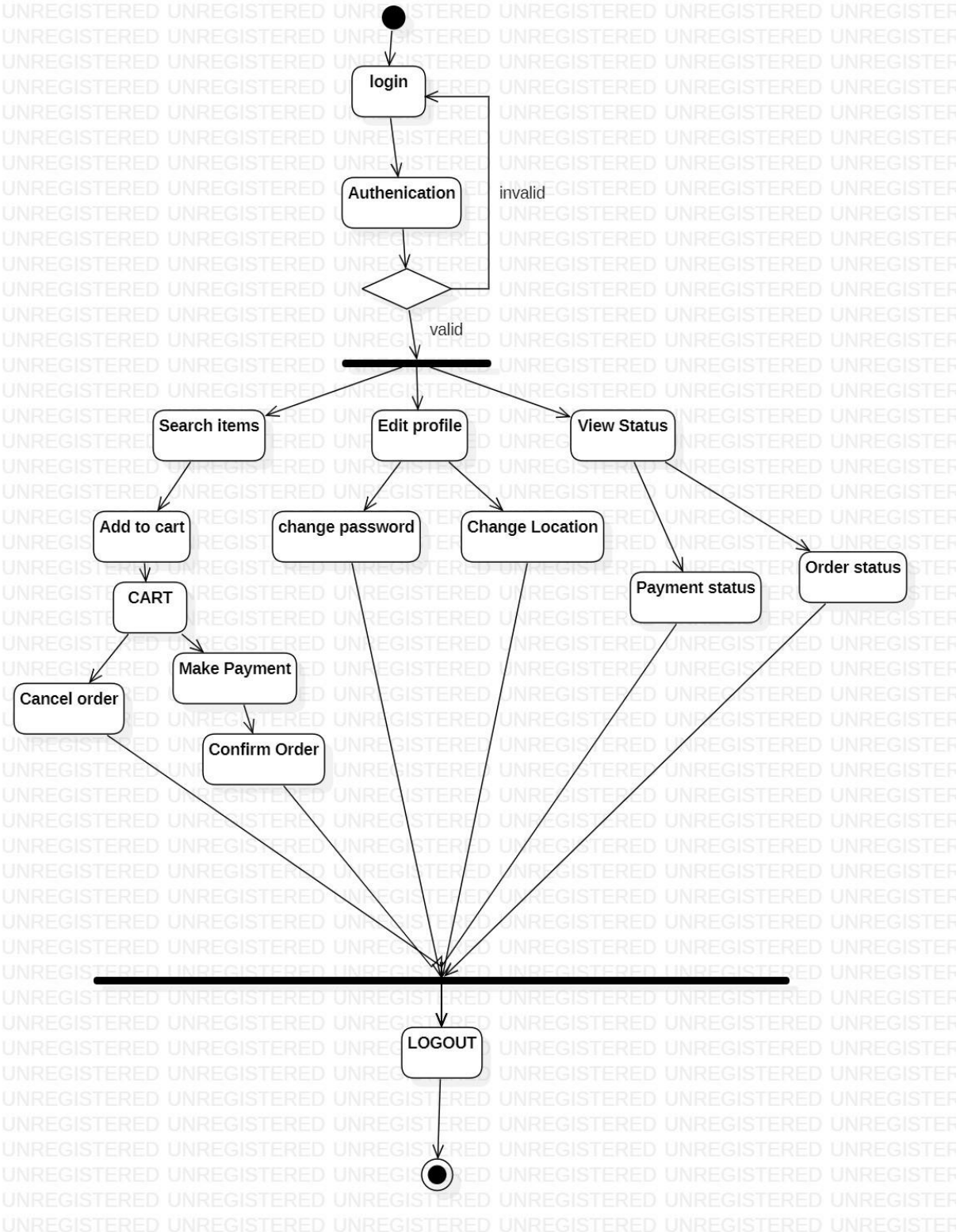


Figure 4.1.5: Activity diagram for user interacting the Flipkart portal

This diagram is the activity diagram of unified modelling language for the web user which shows the flow of activity of order, profile and payment.

The activity diagram for Flipkart includes the features:

a. The user can login to the portal
b. The user can place an order in Flipkart.
c. The user can make payment for the order using types of payment.
d. The user can edit the profiles.
e. The user can check the order and payment status.

SEQUENCE DIAGRAM:

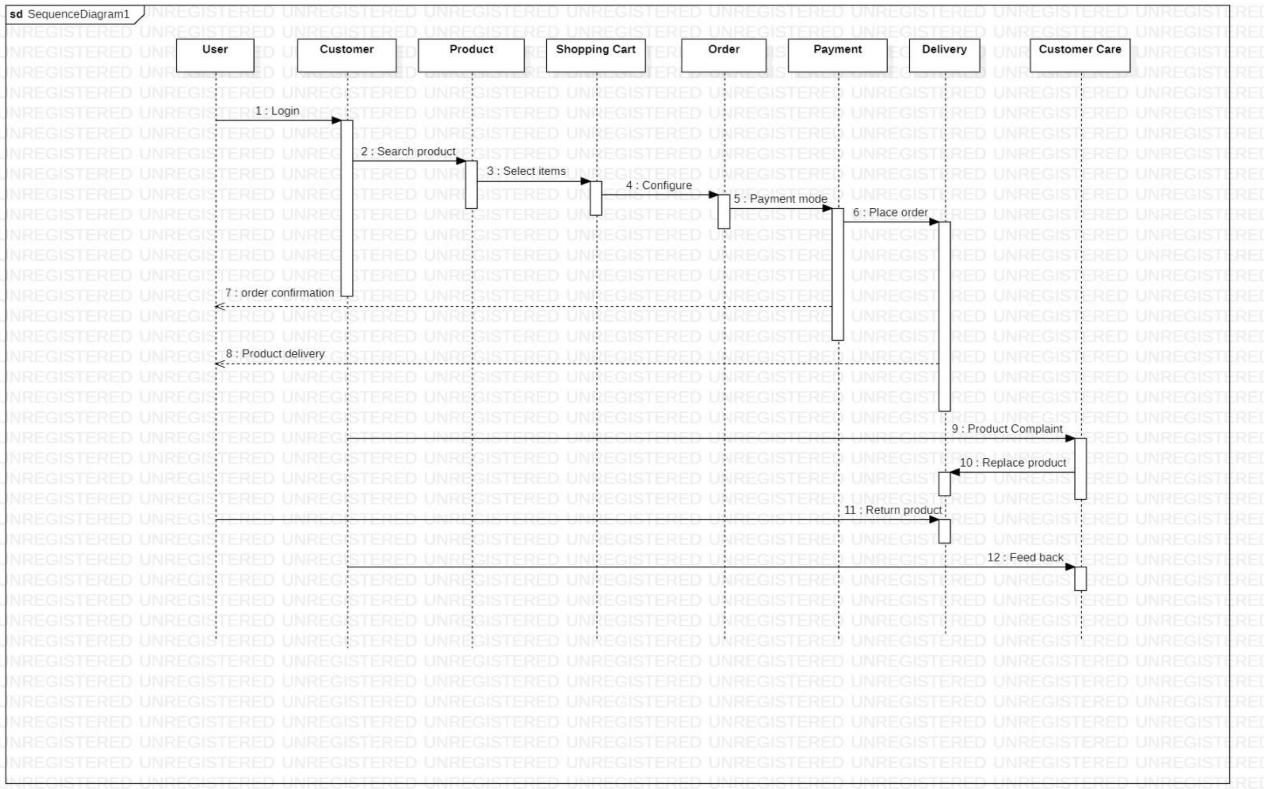


Figure 4.1.6: Sequence diagram for Flipkart

The sequence diagram of unified modelling language for flipkart in which the user login into their existing account using their essential credentials, after logging into the portal the customer can browse vast categories of product in which he can order necessary products at their disposal, the customer can also pay for the products which the Flipkart provides the necessary delivery services as well as the convenient customer care services.

STATE CHART DIAGRAM:

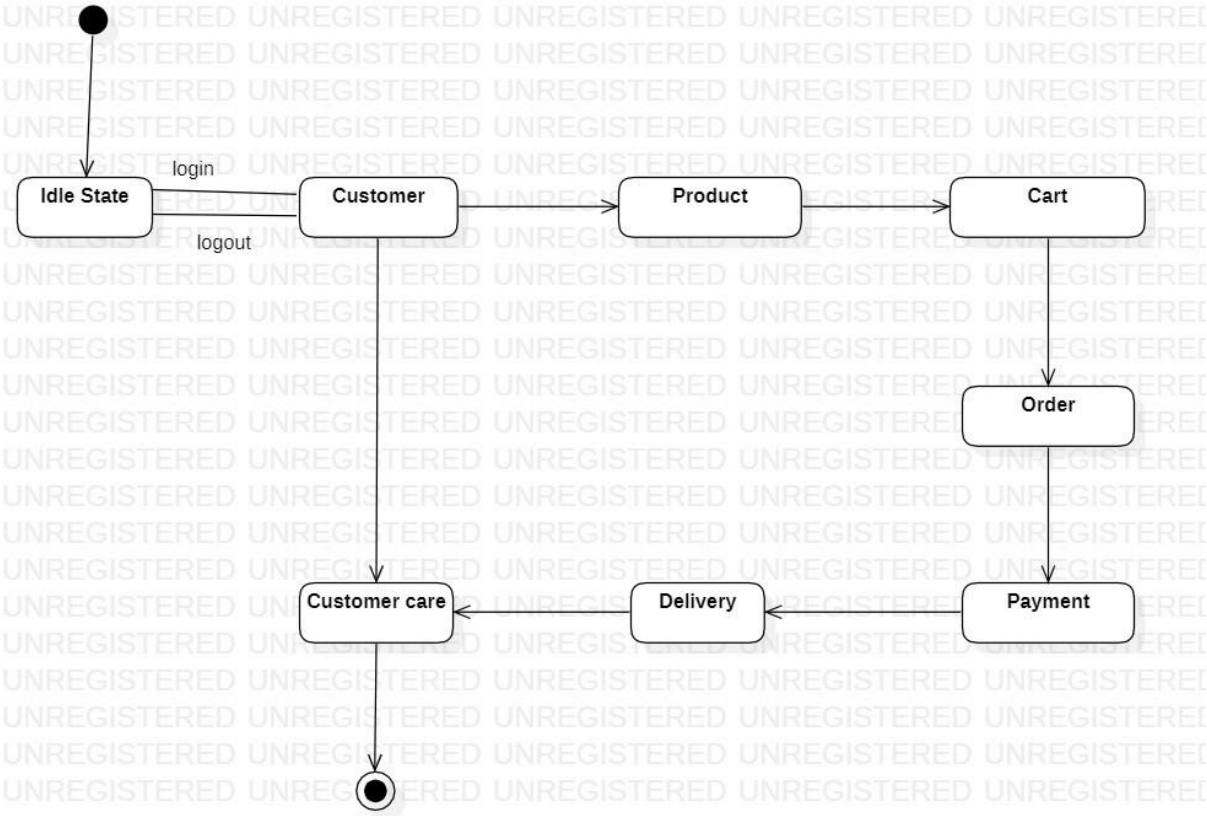


figure 4.1.7: State chart diagram of Fipkart

The state chart diagram of unified modelling language for Flipkart shows the transition of states of web user from idle state to final state.

SYSTEM FLOW DIAGRAMS:

ZERO LEVEL DATA FLOW DIAGRAM

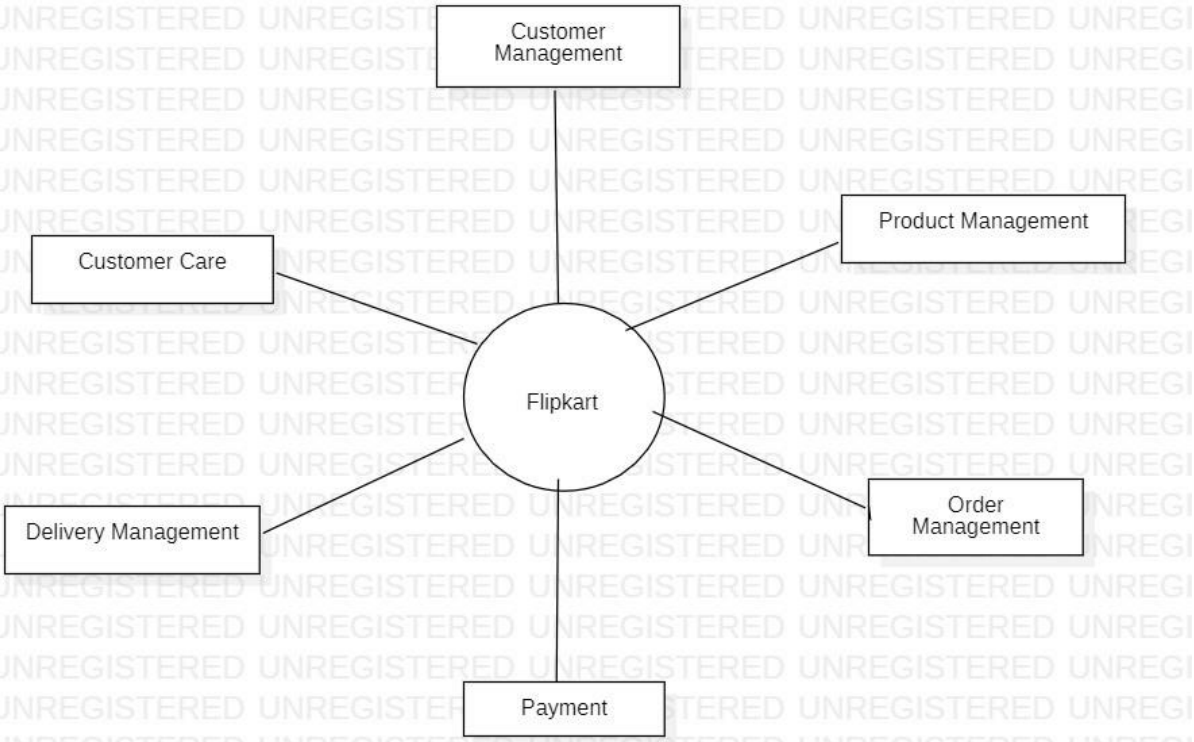


Figure 4.1.8: Data flow diagram level 0

The zero-level data flow diagram of unified modelling language for Flipkart, in which the basic overview of the flipkart ecommerce system is modelled it shows a glance landscape of single high-level processes with its relationship to its respective entities.

High level entities include:

a. Customer management
b. Product Management
c. Order Management
d. Payment
e. Delivery Management
f. Customer Care Management

LEVEL 1 DATA FLOW DIAGRAM:

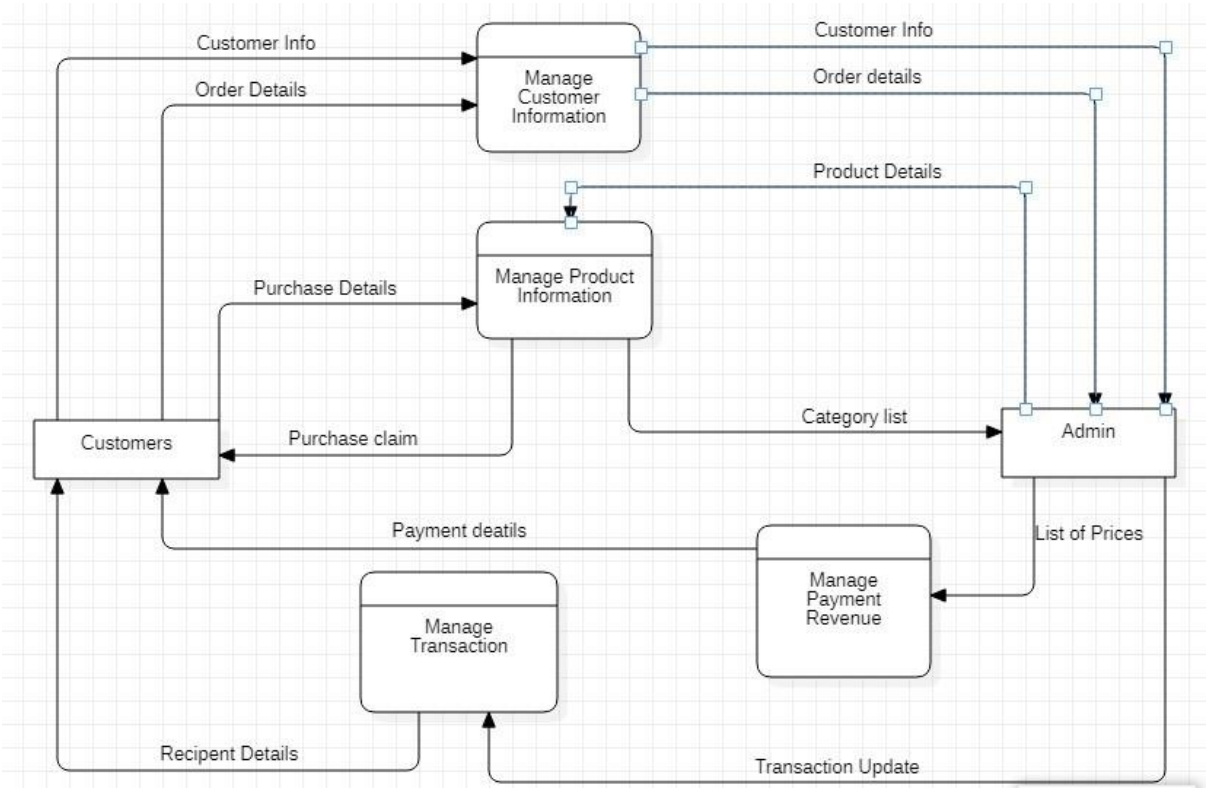


Figure 4.1.9: Data flow diagram level 1

The first -level data flow diagram of unified modelling language for Flipkart in which it represents the system which in turn is divided into system or sub processes where each process deals with one or more data flow transactions from its entities, as they hold provides the required functionality of flipkart.

SWIM LANE DIAGRAM:

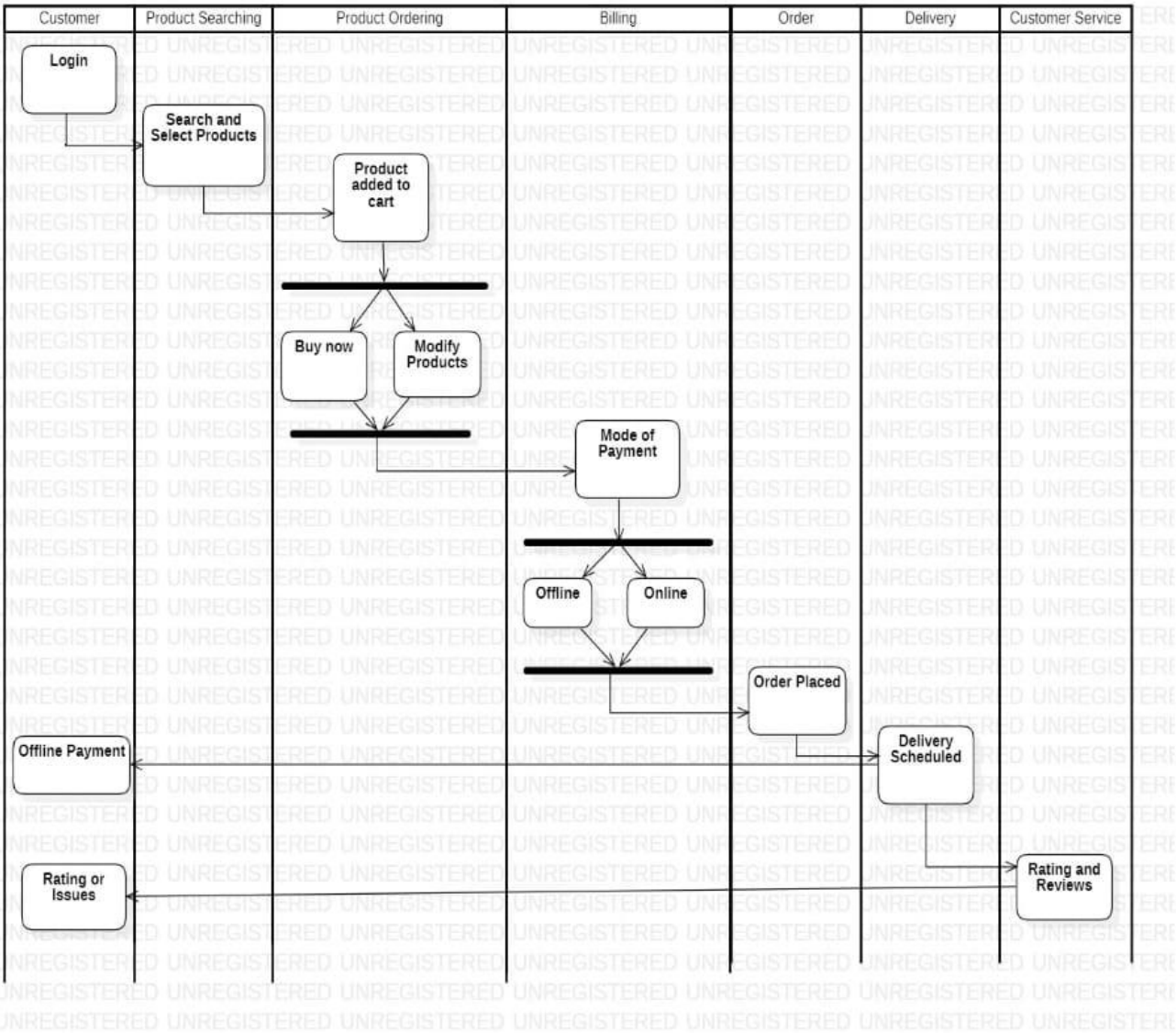


Figure 4.1.10: swimlane diagram

Swimlane diagram describes who is responsible for the activities being performed in the activity diagram and how they are responsible. The activity diagram only represents the activities being performed, but the swimlane describes who does what in a process.

COMPONENT DIAGRAM:

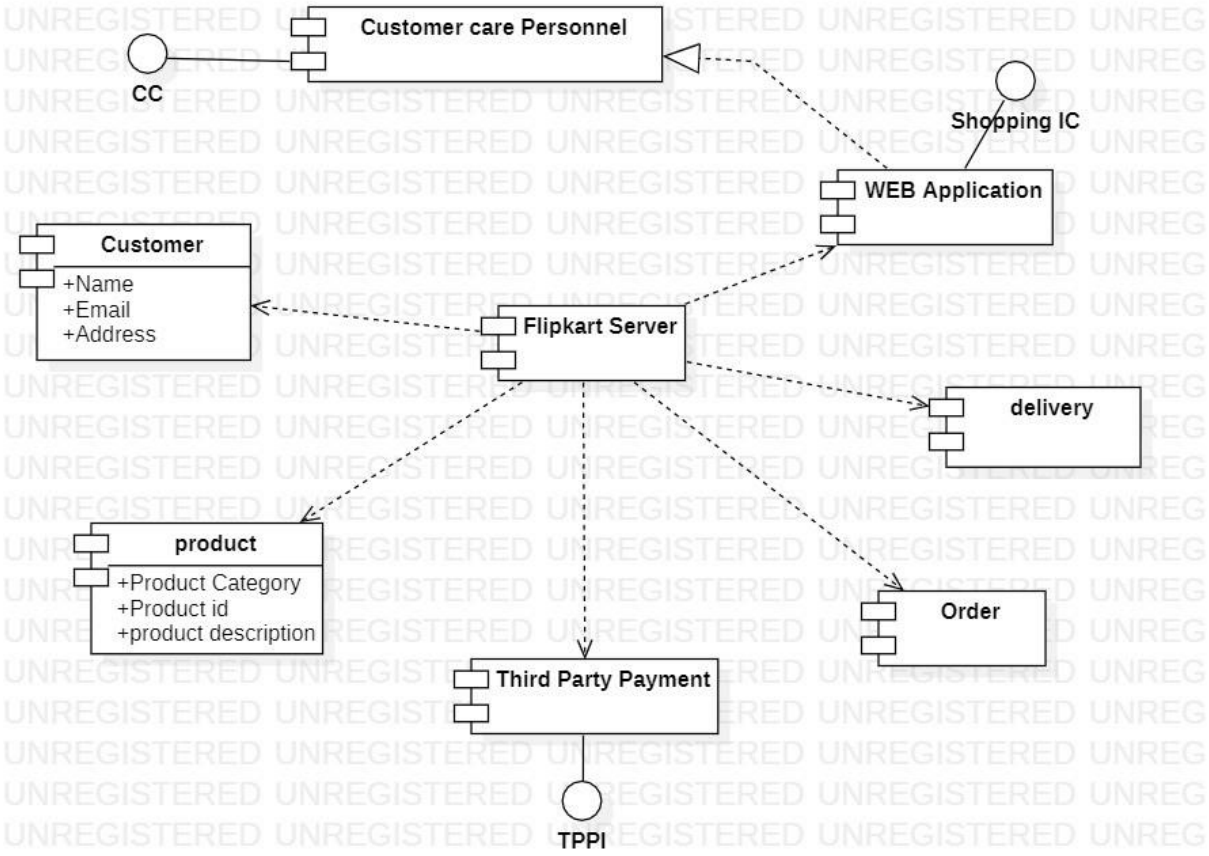


Figure 4.1.11: Component diagram

The component diagram of unified modelling language for Flipkart in which it represents components, ports and relationships between the customer, product, payment, order, delivery, third party payment and customer care. It shows the description of the system as well as relation between various components.

The components of Flipkart include:

a. Customer
b. Product
c. Order
d. Payment
e. Delivery

DEPLOYMENT DIAGRAM:

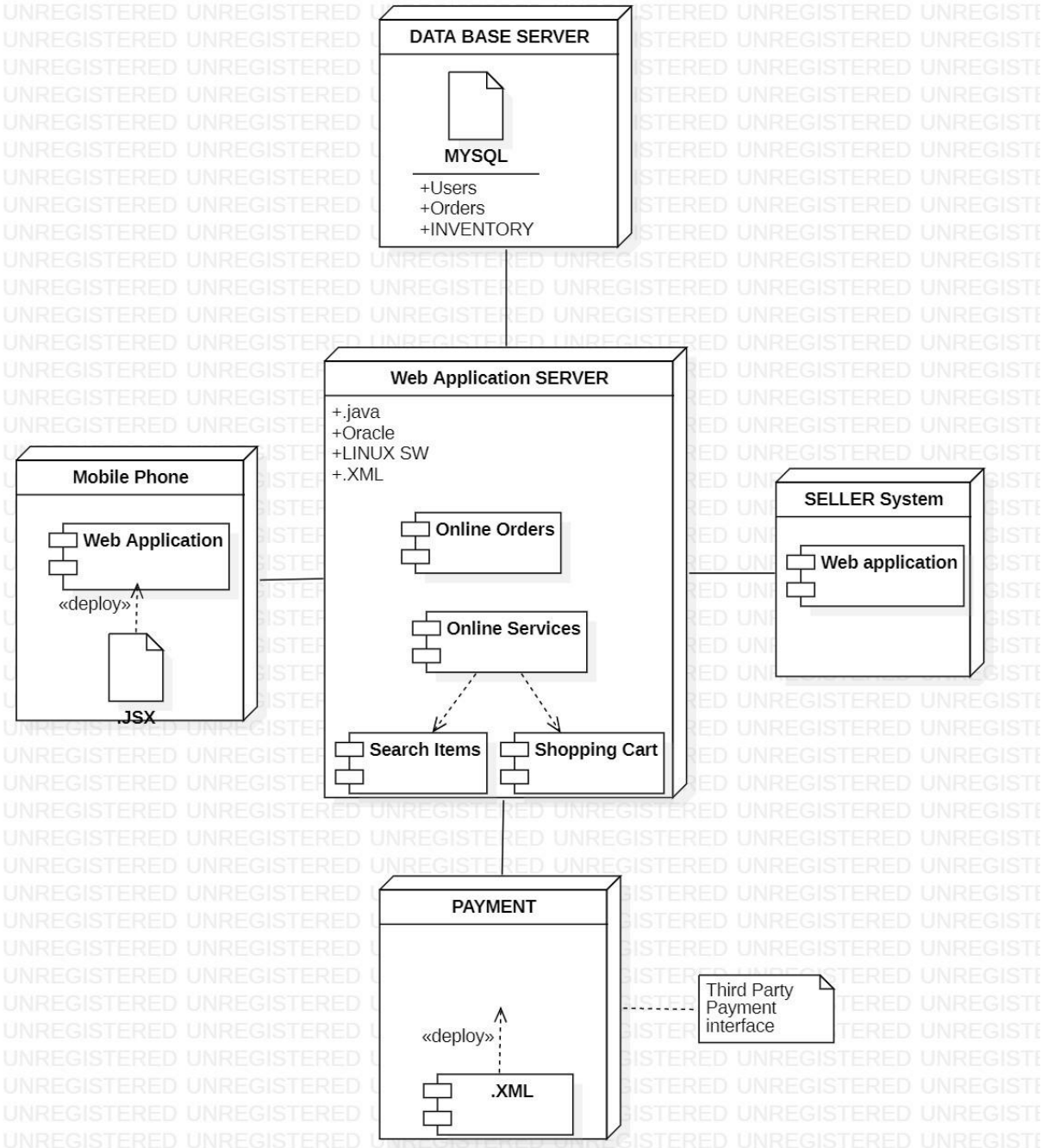


Figure 4.1.12: Deployment diagram

The deployment diagram of unified modelling language for Flipkart in which it provides a view of hardware and software implemented in a system, it also shows various software components deployed in a system.

E-R DIAGRAM

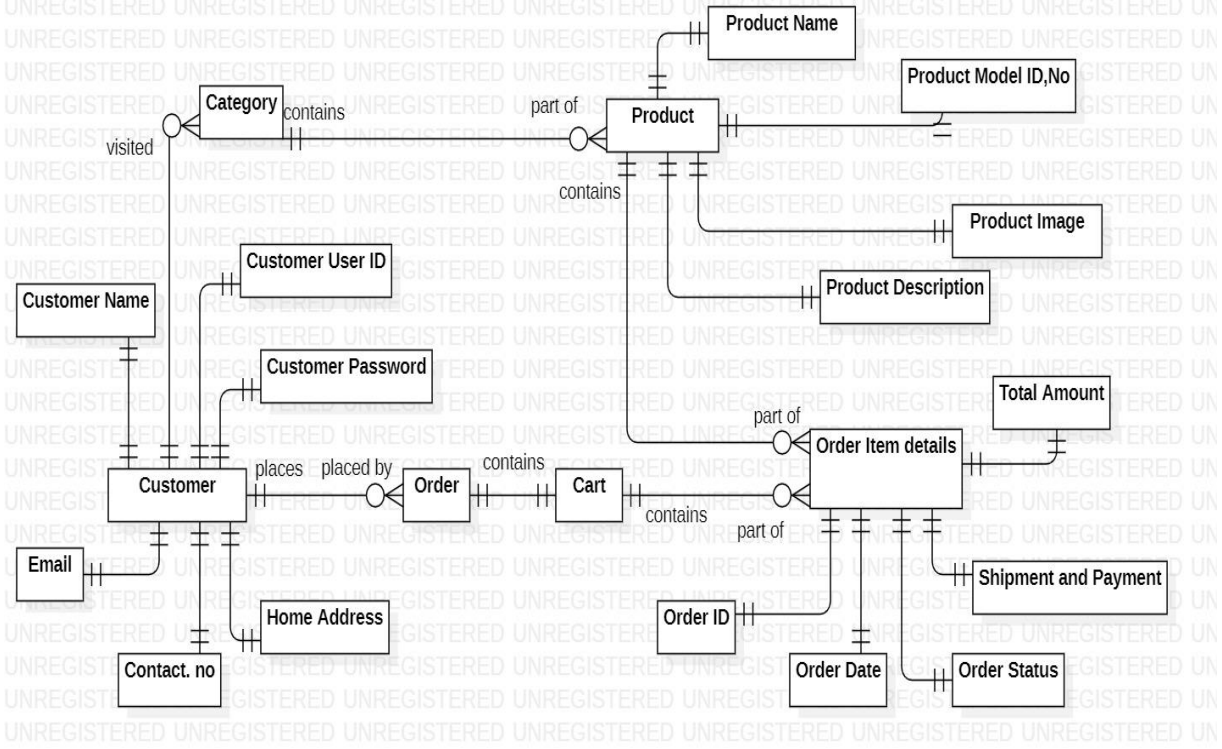


Figure 4.1.13 ER diagram of flipkart

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. ER diagrams are used to sketch out the design of a database

5.IMPLEMENTATION

5.1 Tools Used by Flipkart:

1. Specter
2. Dart Service
3. File Ingest
4. HDFS
5. Dart
6. React Native for iOS and Android
7. Vertica
8. Query UI
9. Android Studio
10. Oracle Erp

5.2 Technologies:

User interface: Apache Lens powered UI.

Server Framework: Java and PHP.

Database: MY SQL

Data Management: Hadoop software

Payment API: Paytm, Phone pe, Google Pay, Pay Zapp, Other UPI payments, Net banking, Debit card payments, Credit card payments.

Security: 256-bit encryption technology

Caching: Memcached.

For Mobile Applications: Java and Kotlin.

PHP:

PHP is a general-purpose scripting language geared towards web development. It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994. The PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response.

Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside the web context, such as standalone graphical applications and robotic drone control code can also be directly executed from the command line.

MY SQL:

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often, MySQL is used with other programs to implement applications that need relational database capability. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress. MySQL is also used by many popular websites, including Facebook, Flipkart, Flickr, MediaWiki, Twitter, and YouTube.

Hadoop:

Apache Hadoop Is a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model. Hadoop was originally designed for computer clusters built from commodity hardware, which is still the common use. It has since also found use on clusters of higher-end hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality, where nodes manipulate the data, they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

Memcached:

Memcached (pronounced variously *mem-cash-dee* or *mem-cashed*) is a general-purpose distributed memory-caching system. It is often used to speed up dynamic database-driven websites by caching data and objects in RAM to reduce the number of times an external data source (such as a database or API) must be read. Memcached is free and open-source software, licensed under the Revised BSD license. Memcached runs on Unix-like operating systems (Linux and macOS) And on Microsoft Windows. It depends on the libevent library.

Memcached's APIs provide a very large hash table distributed across multiple machines. When the table is full, subsequent inserts cause older data to be purged in least recently used (LRU) order. Applications using Memcached typically layer requests and additions into RAM before falling back on a slower backing store, such as a database.

JAVA:

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need to recompile. applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client–server web applications, with a reported 9 million developers.

KOTLIN:

Kotlin is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of Kotlin's standard library depends on the Java Class Library, but type inference allows its syntax to be more concise. Kotlin mainly targets the JVM, but also compiles to JavaScript (e.g., for frontend web applications using React) Or native code via LLVM (e.g., for native iOS apps sharing business logic with Android apps). Language development costs are borne by JetBrains, while the Kotlin Foundation protects the Kotlin trademark.

React Native:

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, react is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

Apache Lens Powered UI:

Lens provides a Unified Analytics interface. Lens aims to cut the Data Analytics silos by providing a single view of data across multiple tiered data stores and optimal execution environment for the analytical query. It seamlessly integrates Hadoop with traditional data warehouses to appear like one.

5.3 SAMPLE CODE

```
<?php
/**
* Demo Code
* PHP Wrapper for Flipkart API
* License: MIT License
*/

//This is basic example code, and is not intended to be used in production.
```

```
//Don't forget to use a valid Affiliate Id and Access Token.

//Include the class.
include "clusterdev.flipkart-api.php";

//Replace <affiliate-id> and <access-token> with the correct values
$flipkart = new \clusterdev\Flipkart("<affiliate-id>", "<access-token>", "json");

$dotd_url = 'https://affiliate-api.flipkart.net/affiliate/offers/v1/dotd/json';
$topoffers_url = 'https://affiliate-api.flipkart.net/affiliate/offers/v1/top/json';

//To view category pages, API URL is passed as query string.
$url = isset($_GET['url'])?$_GET['url']:false;
if($url){
	//URL is base64 encoded to prevent errors in some server setups.
	$url = base64_decode($url);

	//This parameter lets users allow out-of-stock items to be displayed.
	$hidden = isset($_GET['hidden'])?false:true;

	//Call the API using the URL.
	$details = $flipkart->call_url($url);
	if(!$details){
		echo 'Error: Could not retrieve products list.';
		exit();
	}

	//The response is expected to be JSON. Decode it into associative arrays.
	$details = json_decode($details, TRUE);
	//The response is expected to contain these values.
	$nextUrl = $details['nextUrl'];
	$validTill = $details['validTill'];
	$products = $details['productInfoList'];
```

```
        //The navigation buttons.
        echo '<h2><a href="?">HOME</a> | <a
href="?url='.base64_encode($nextUrl).'">NEXT >></a></h2>';

        //Message to be displayed if out-of-stock items are hidden.
        if($hidden)
                echo 'Products that are out of stock are hidden by default.<br><a
href="?hidden=1&url='.base64_encode($url).'">SHOW OUT-OF-STOCK
ITEMS</a><br><br>';

        //Products table
        echo "<table border=2 cellpadding=10 cellspacing=1 style='text-align:center'>";
        $count = 0;
        $end = 1;
        //Make sure there are products in the list.
        if(count($products) > 0){
                foreach ($products as $product) {

                        //Hide out-of-stock items unless requested.
                        $inStock = $product['productBaseInfo']['productAttributes']['inStock'];
                        if(!$inStock && $hidden)
                                continue;

                        //Keep count.
                        $count++;

                        //The API returns these values nested inside the array.
                        //Only image, price, url and title are used in this demo
                        $productId =
$product['productBaseInfo']['productIdentifier']['productId'];
                        $title = $product['productBaseInfo']['productAttributes']['title'];
                        $productDescription =
$product['productBaseInfo']['productAttributes']['productDescription'];
```

```
                    //We take the 200x200 image, there are other sizes too.
                    $productImage = array_key_exists('200x200',
$product['productBaseInfo']['productAttributes']['imageUrls'])?$product['productBaseInfo']['p
roductAttributes']['imageUrls']['200x200']:'';
                    $sellingPrice =
$product['productBaseInfo']['productAttributes']['sellingPrice']['amount'];
                    $productUrl =
$product['productBaseInfo']['productAttributes']['productUrl'];
                    $productBrand =
$product['productBaseInfo']['productAttributes']['productBrand'];
                    $color = $product['productBaseInfo']['productAttributes']['color'];
                    $productUrl =
$product['productBaseInfo']['productAttributes']['productUrl'];

                    //Setting up the table rows/columns for a 3x3 view.
                    $end = 0;
                    if($count%3==1)
                            echo '<tr><td>';
                    else if($count%3==2)
                            echo '</td><td>';
                    else{
                            echo '</td><td>';
                            $end =1;
                    }

                    echo '<a target="_blank" href="'.$productUrl.'"><img
src="'.$productImage.'"/><br>'.$title."</a><br>Rs. ".$sellingPrice;

                    if($end)
                            echo '</td></tr>';
            }
    }

    //A message if no products are printed.
```

```
	if($count==0){
		echo '<tr><td>The retrieved products are not in stock. Try the Next button or another category.</td><tr>';
	}

	//A hack to make sure the tags are closed.
	if($end!=1)
		echo '</td></tr>';

	echo '</table>';

	//Next URL link at the bottom.
	echo '<h2><a href="?url='.base64_encode($nextUrl).'">NEXT >></a></h2>';

	//That's all we need for the category view.
	exit();
}


//Deal of the Day DOTD and Tops offers
$offer = isset($_GET['offer'])?$_GET['offer']:false;
if($offer){

	if($offer == 'dotd'){
		//Call the API using the URL.
		$details = $flipkart->call_url($dotd_url);

		if(!$details){
			echo 'Error: Could not retrieve DOTD.';
			exit();
		}

		//The response is expected to be JSON. Decode it into associative arrays.
		$details = json_decode($details, TRUE);
```

```
		$list = $details['dotdList'];

		//The navigation buttons.
		echo '<h2><a href="?">HOME</a> | DOTD Offers | <a
href="?offer=topoffers">Top Offers</a></h2>';

		//Show table
		echo "<table border=2 cellpadding=10 cellspacing=1 style='text-
align:center'>";
		$count = 0;
		$end = 1;

		//Make sure there are products in the list.
		if(count($list) > 0){
			foreach ($list as $item) {
				//Keep count.
				$count++;

				//The API returns these values
				$title = $item['title'];
				$description = $item['description'];
				$url = $item['url'];
				$imageUrl = $item['imageUrls'][0]['url'];
				$availability = $item['availability'];

				//Setting up the table rows/columns for a 3x3 view.
				$end = 0;
				if($count%3==1)
					echo '<tr><td>';
				else if($count%3==2)
					echo '</td><td>';
				else{
					echo '</td><td>';
```

```
                                $end =1;
                        }

                        echo '<a target="_blank" href="'.$url.'"><img
src="'.$imageUrl.'" style="max-width:200px; max-
height:200px;"/><br>'.$title."</a><br>".$description;

                        if($end)
                                echo '</td></tr>';

                }
        }
        //A message if no products are printed.
        if($count==0){
                echo '<tr><td>No DOTDs returned.</td><tr>';
        }

        //A hack to make sure the tags are closed.
        if($end!=1)
                echo '</td></tr>';

        echo '</table>';

        //That's all we need for the category view.
        exit();
}else if($offer == 'topoffers'){

        //Call the API using the URL.
        $details = $flipkart->call_url($topoffers_url);

        if(!$details){
                echo 'Error: Could not retrieve Top Offers.';
                exit();
        }
```

```
		//The response is expected to be JSON. Decode it into associative arrays.
		$details = json_decode($details, TRUE);

		$list = $details['topOffersList'];

		//The navigation buttons.
		echo '<h2><a href="?">HOME</a> | <a href="?offer=dotd">DOTD Offers</a> | Top Offers</h2>';

		//Show table
		echo "<table border=2 cellpadding=10 cellspacing=1 style='text-align:center'>";
		$count = 0;
		$end = 1;

		//Make sure there are products in the list.
		if(count($list) > 0){
			foreach ($list as $item) {
				//Keep count.
				$count++;

				//The API returns these values
				$title = $item['title'];
				$description = $item['description'];
				$url = $item['url'];
				$imageUrl = $item['imageUrls'][0]['url'];
				$availability = $item['availability'];

				//Setting up the table rows/columns for a 3x3 view.
				$end = 0;
				if($count%3==1)
					echo '<tr><td>';
				else if($count%3==2)
```

```
                    echo '</td><td>';
                else{
                    echo '</td><td>';
                    $end =1;
                }

                echo '<a target="_blank" href="'.$url.'"><img
src="'.$imageUrl.'" style="max-width:200px; max-
height:200px;"/><br>'.$title."</a><br>".$description;

                if($end)
                    echo '</td></tr>';

            }
        }
        //A message if no products are printed.
        if($count==0){
            echo '<tr><td>No Top Offers returned.</td><tr>';
        }

        //A hack to make sure the tags are closed.
        if($end!=1)
            echo '</td></tr>';

        echo '</table>';

        //That's all we need for the category view.
        exit();

    }else{
        echo 'Error: Invalid offer type.';
        exit();
    }
```

```
}


//If the control reaches here, the API directory view is shown.

//Query the API
$home = $flipkart->api_home();

//Make sure there is a response.
if($home==false){
	echo 'Error: Could not retrieve API homepage';
	exit();
}

//Convert into associative arrays.
$home = json_decode($home, TRUE);

$list = $home['apiGroups']['affiliate']['apiListings'];

echo '<h1>API Homepage</h1><h2><a href="?offer=dotd">DOTD Offers</a> | <a
href="?offer=topoffers">Top Offers</a></h2>Click on a category link to show available
products from that category.<br><br>';

//Create the tabulated view for different categories.
echo '<table border=2 style="text-align:center;">';
$count = 0;
$end = 1;
foreach ($list as $key => $data) {
	$count++;
	$end = 0;

	//To build a 3x3 table.
	if($count%3==1)
		echo '<tr><td>';
```

```
        else if($count%3==2)
                echo '</td><td>';
        else{
                echo '</td><td>';
                $end =1;
        }

        echo "<strong>".$key."</strong>";
        echo "<br>";
        //URL is base64 encoded when sent in query string.
        echo '<a
href="?url='.base64_encode($data['availableVariants']['v0.1.0']['get']).'">View Products
&raquo;</a>';
}

if($end!=1)
        echo '</td></tr>';
echo '</table>';
```

6.TESTING

6.1 Types of testing:

Unit Testing:

Unit Testing is a software testing technique by means of which individual units of software i.e, group of computer program modules, usage procedures and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which is very independent modules are tested to determine if there are any issue by the developer himself. It is correlated with functional correctness of the independent modules.

Unit Testing is defined as a type of software testing where individual components of a software tested. Unit Testing of software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer.

In this project we use unit testing to verify the flipkart application in development phase, and in this testing the modules which are specifically required for the flipkart application project are tested through unit testing.

The main modules include:

1. Customer operation system
2. Product management system
3. Order management system
4. Payment operation system
5. Delivery logistics
6. Customer Care management

Integration Testing:

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Integration testing ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly. In this testing the modules such as customer, payment, order, delivery, and customer care is integrated with respect to operation such as login, browse, ordering, account handling, payment and shipping in this respective integration testing.

Validation testing:

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e, it checks what we are developing is the right product. it is validation of actual product.
Validation is the Dynamic Testing.

In this testing phase the individual modules are verified and validated with their respective operations for functionalities there present by these upcoming next testing phases can be performed conveniently e.g.: the payment system provides different payment such as offline and online mode as the test scenarios is passed the test is successfully completed.

System Testing:
System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.

In This phase all the system modules with the whole system tests for debugging desire issues, interface issues, software and hardware complications, error handling etc.

6.2 TEST CASE

6.2.1 Product functionality of Flipkart

Test Case ID	Module Name	Test Scenario	Test Case	Pre-requisites	Test Step	Expected Result
TC001	Product Details	Verify the details on Product Specification page	Verify that images of product are displayed correctly.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on a product displayed on home page.	Browser should be opened. Ecommerce website should be opened. user should be able to input username and password the images of product should be displayed correctly.
TC002	Product Details	Verify the details on Product Specification page	Verify that price of product is displayed.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on a product displayed on home page.	Browser should be opened. Ecommerce website should be opened. user should be able to input username and password the images of product should be displayed correctly.

TC003	Product Details	Verify the details on Product Specification page	Verify that product reviews are mentioned.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on a product displayed on home page.	Browser should be opened. Ecommerce website should be opened. user should be able to input username and password the images of product should be displayed correctly.
TC004	Product Details	Verify the details on Product Specification page	Verify that product specifications are displayed.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on a product displayed on home page.	Browser should be opened. Ecommerce website should be opened. user should be able to input username and password the images of product should be displayed correctly.
TC005	Product Details	Verify the details on Product Specification page	Verify that information about IN-Stock/Out-Stock are displayed.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button	Browser should be opened. Ecommerce website should be opened. user should be able to input

					5. Click on a product displayed on home page.	username and password the images of product should be displayed correctly.

Table 6.2.1: test case for product functionality of flipkart

6.2 Order functionality of Flipkart:

Test Case ID	Module Name	Test Scenario	Test Case	Pre-requisites	Test Step	Expected Result
TC001	My orders page	Verify the details on Orders Page	Verify that user should be able to track the order on My orders page.	Browser should be installed Internet connection should be present User should be logged in	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5.Click on My orders button 6.Click on Track order for an order.	Browser should be opened Ecommerce website should be opened user should be able to input username and password Home page should be displayed after login The orders by user should be displayed. Tracking information should be displayed for that order.
TC002	My orders page	Verify the details on Orders Page	Verify that user should be able to change the	Browser should be installed Internet connection	1.Open browser 2.Open URL	Browser should be opened Ecommerce website should be opened

			delivery date and time.	should be present User should be logged in	3.Enter username and password 4.Click on login button 5.Click on My orders button 6. Click on change delivery date and time button.	user should be able to input username and password Home page should be displayed after login The orders by user should be displayed. Tracking information should be displayed for that order.
TC003	My orders page	Verify the details on Orders Page	Verify that user should be able to cancel the order	Browser should be installed Internet connection should be present User should be logged in	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5.Click on My orders button 6. Click on cancel order for an order.	Browser should be opened Ecommerce website should be opened user should be able to input username and password Home page should be displayed after login The orders by user should be displayed. Tracking information should be displayed for that order.
TC004	My orders page	Verify the details on Orders Page	Verify that user should be able to return the order after	Browser should be installed Internet connection	1.Open browser 2.Open URL 3.Enter username and password	Browser should be opened Ecommerce website should be opened

			delivery of order.	should be present User should be logged in	4.Click on login button 5.Click on My orders button 6. Click on return item for an order.	user should be able to input username and password Home page should be displayed after login The orders by user should be displayed. Tracking information should be displayed for that order.
TC005	My orders page	Verify the details on Orders Page	Verify that user should be able to exchange the order from My orders page.	Browser should be installed Internet connection should be present User should be logged in	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5.Click on My orders button 6. Click on exchange item for an order.	Browser should be opened Ecommerce website should be opened user should be able to input username and password Home page should be displayed after login The orders by user should be displayed. Tracking information should be displayed for that order.

Table 6.2.2: test case for order functionality of flipkart

6.3 Payment functionality of Flipkart

Test Case	Module scenario	Test Scenario	Test case	Pre-requisite	Test step	Expected Result
TC001	Checkout, Payments Page	Verify the details on Checkout, Payments Page	Verify that payments options applicable for the order should be displayed at checkout.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on any product displayed on home page 6. Click on add to cart for the product. 7. Click on Checkout button.	Browser should be opened Ecommerce website should be opened user should be able to input username and password Home page should be displayed after login The product page should be displayed. The product should be added to cart. The checkout page should be displayed with payments options.
TC002	Checkout, Payments Page	Verify the details on Checkout, Payments Page	Verify that delivery details of items should be displayed at checkout.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on any product displayed on home page 6. Click on add to cart for the product.	Browser should be opened Ecommerce website should be opened user should be able to input username and password Home page should be displayed after login The product page should be displayed. The product should be added to cart.

					7. Click on Checkout button.	The checkout page should be displayed with payments options.
TC003	Checkout, Payments Page	Verify the details on Checkout, Payments Page	Verify that user should get order details by message or email.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on any product displayed on home page 6. Click on add to cart for the product. 7. Click on Checkout button.	Browser should be opened Ecommerce website should be opened user should be able to input username and password Home page should be displayed after login The product page should be displayed. The product should be added to cart. The checkout page should be displayed with payments options.
TC004	Checkout, Payments Page	Verify the details on Checkout, Payments Page	Verify that user should be directed to home page after checkout.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on any product displayed on home page 6. Click on add to cart for the product.	Browser should be opened Ecommerce website should be opened user should be able to input username and password Home page should be displayed after login The product page should be displayed.

					7. Click on Checkout button.	The product should be added to cart. The checkout page should be displayed with payments options.
TC005	Checkout, Payments Page	Verify the details on Checkout, Payments Page	Verify that user should be displayed payment options such as Credit Card, Debit Card, Net Banking etc.	Browser should be installed Internet connection should be present	1.Open browser 2.Open URL 3.Enter username and password 4.Click on login button 5. Click on any product displayed on home page 6. Click on add to cart for the product. 7. Click on Checkout button.	Browser should be opened Ecommerce website should be opened user should be able to input username and password Home page should be displayed after login The product page should be displayed. The product should be added to cart. The checkout page should be displayed with payments options.

Table 6.2.3: test case for payment functionality of flipkart

RESULT: The Flipkart ecommerce system was designed and implemented successfully.

6.3 USER INTERFACES

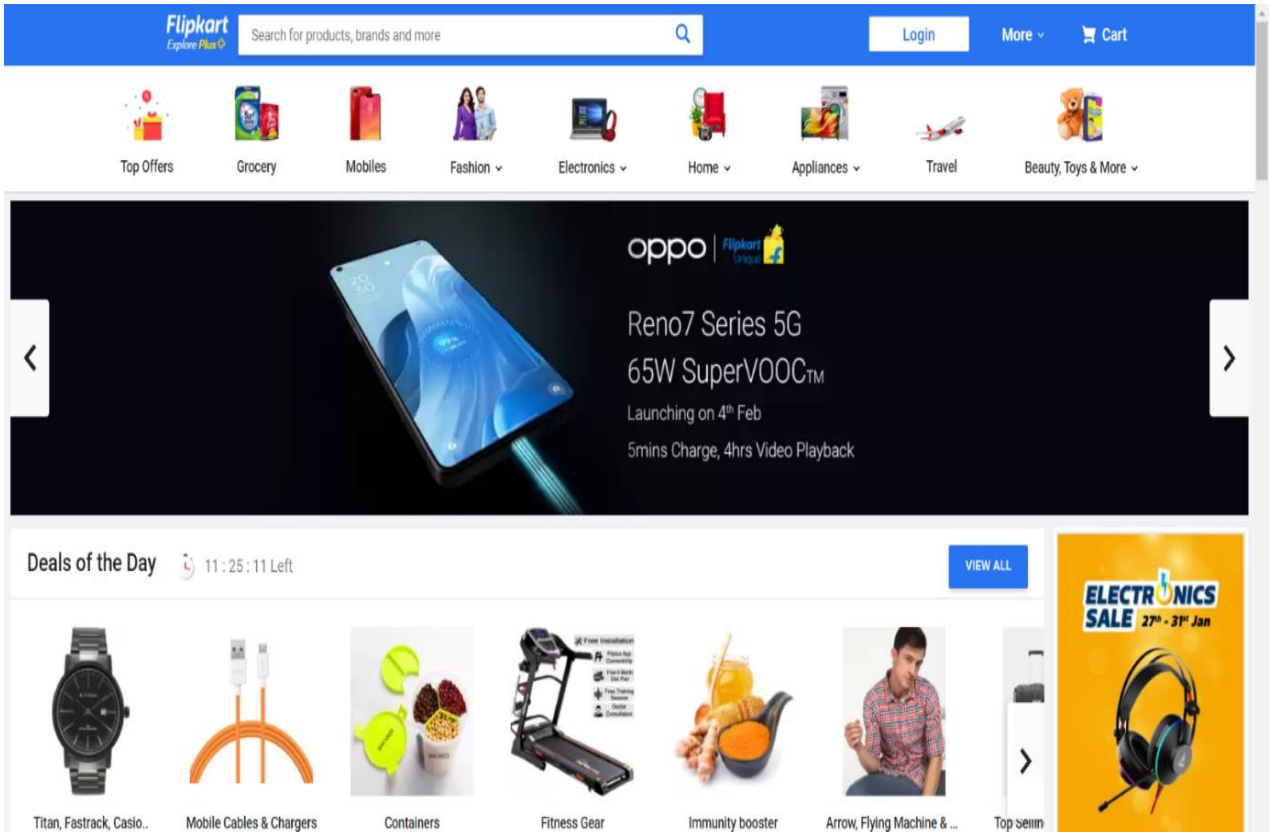


Figure 6.3.1 homepage

Home page:

In this we can browse, access to catalog and see exciting deals.

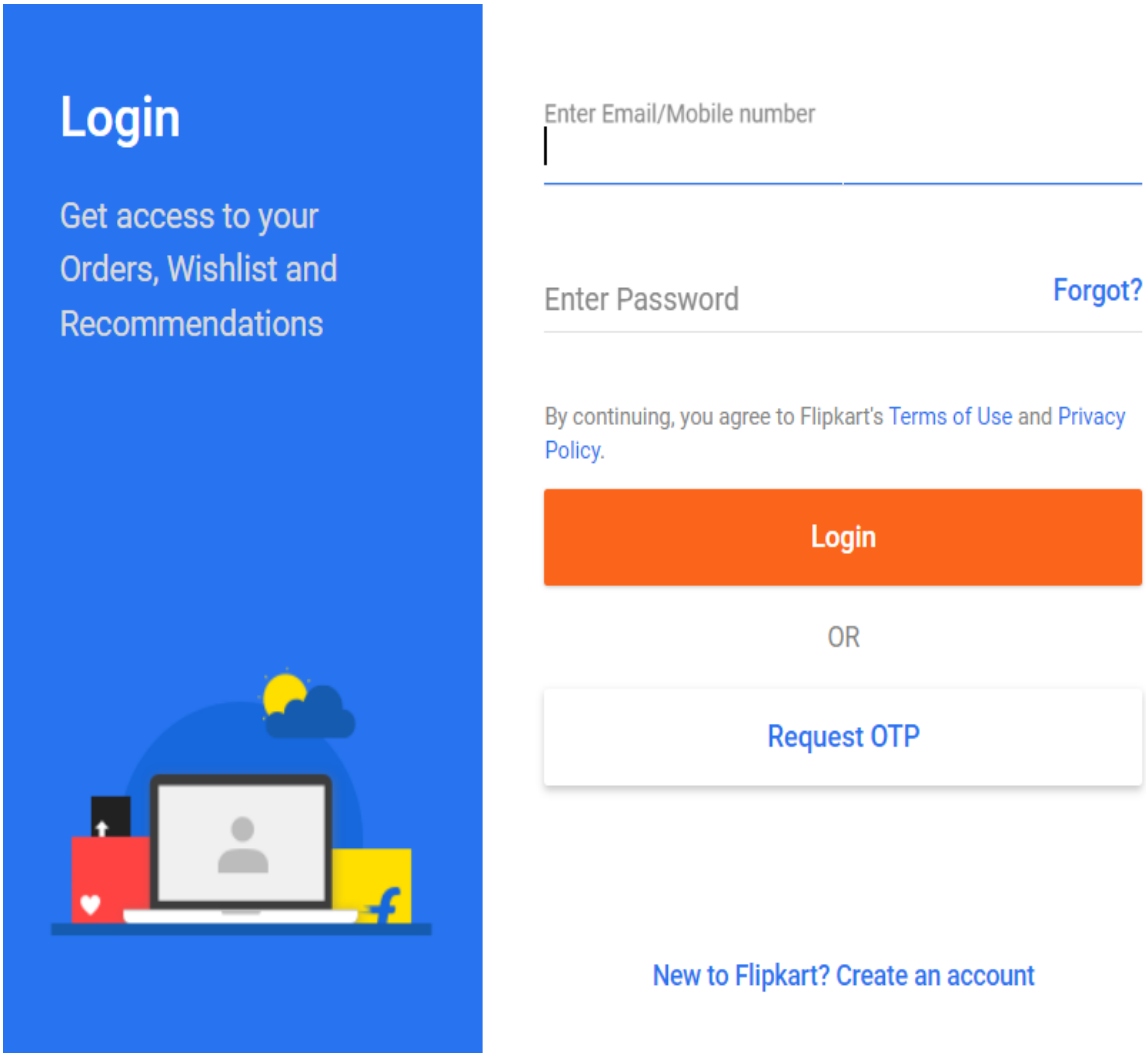


Figure 6.3.2 login page

Login page:

In this page the user has to authenticate with his respective user id and password.

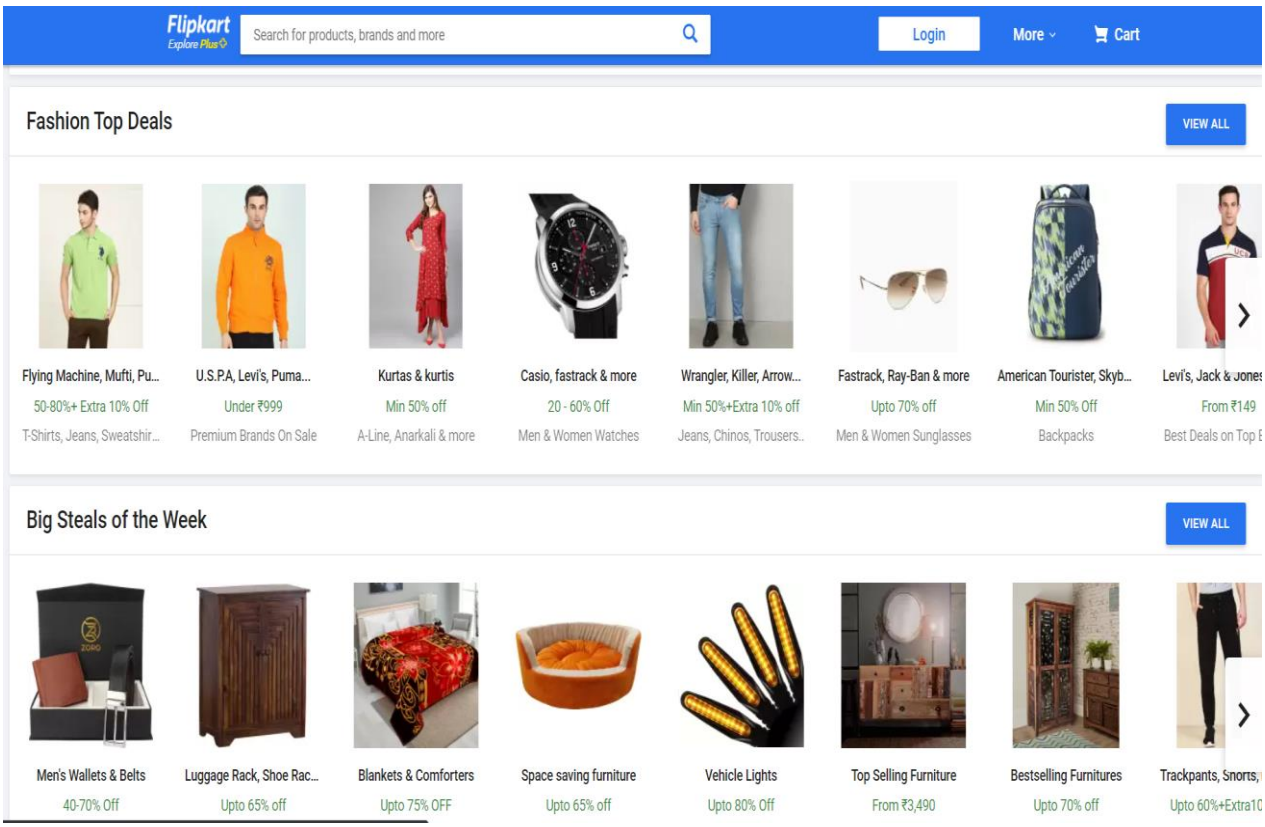


Figure 6.3.3 deals of the day page

Deals of the day page:

In this page we can view deals of the day and exciting offers.

Figure 6.3.4 categories page

Categories page:

In this page we can find variety and category of products

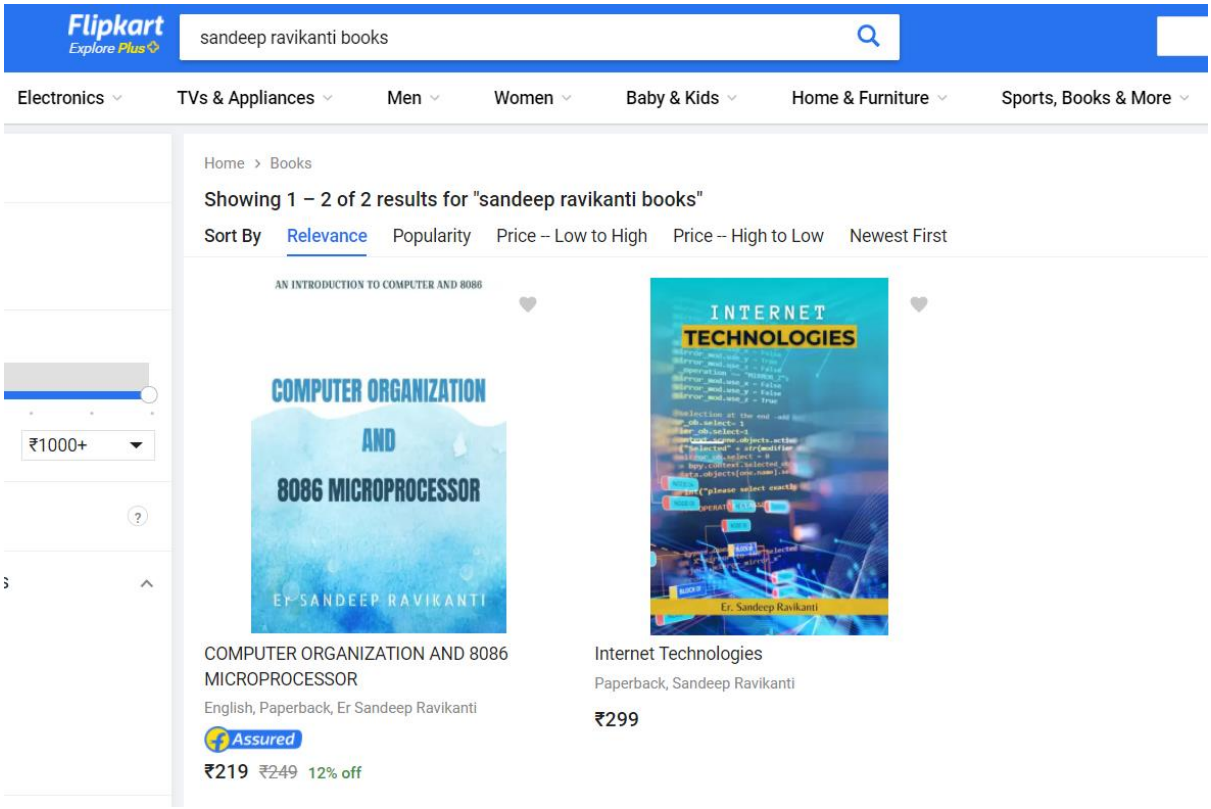


Figure 6.3.5 search engine page

Search engine:

In this page we can search required products for the user's specific needs.

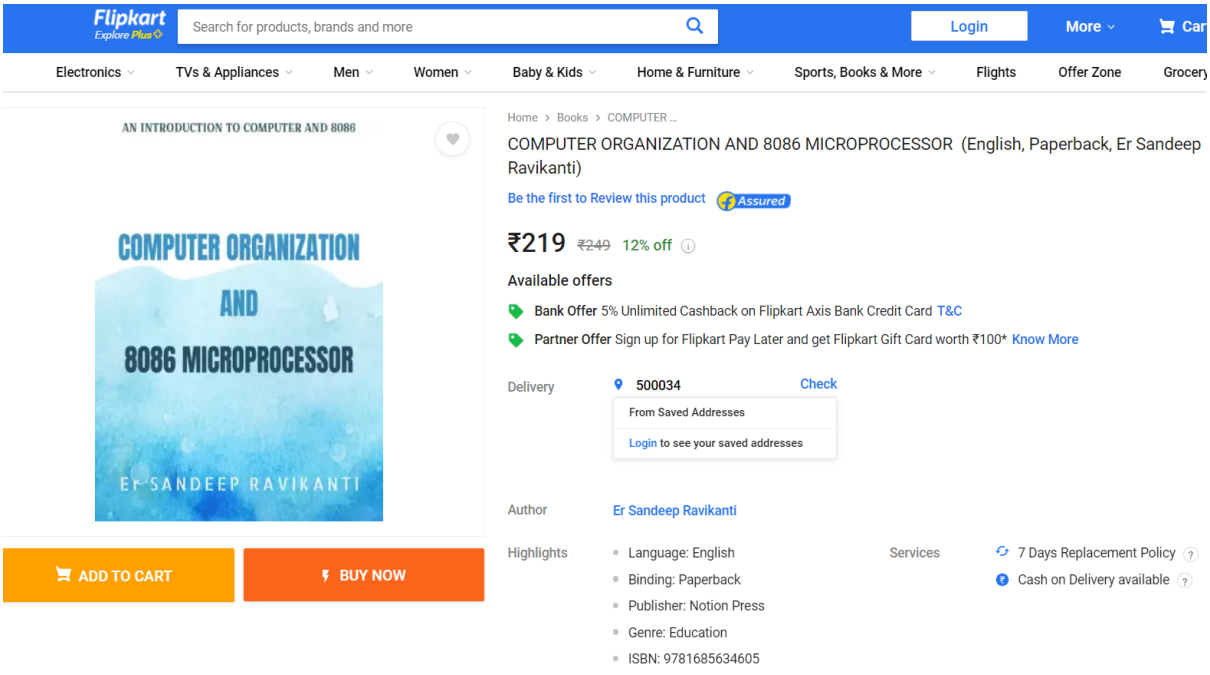


Figure 6.3.6 order page

Order page:

In this page we can view the order description, order price and order feasibility.

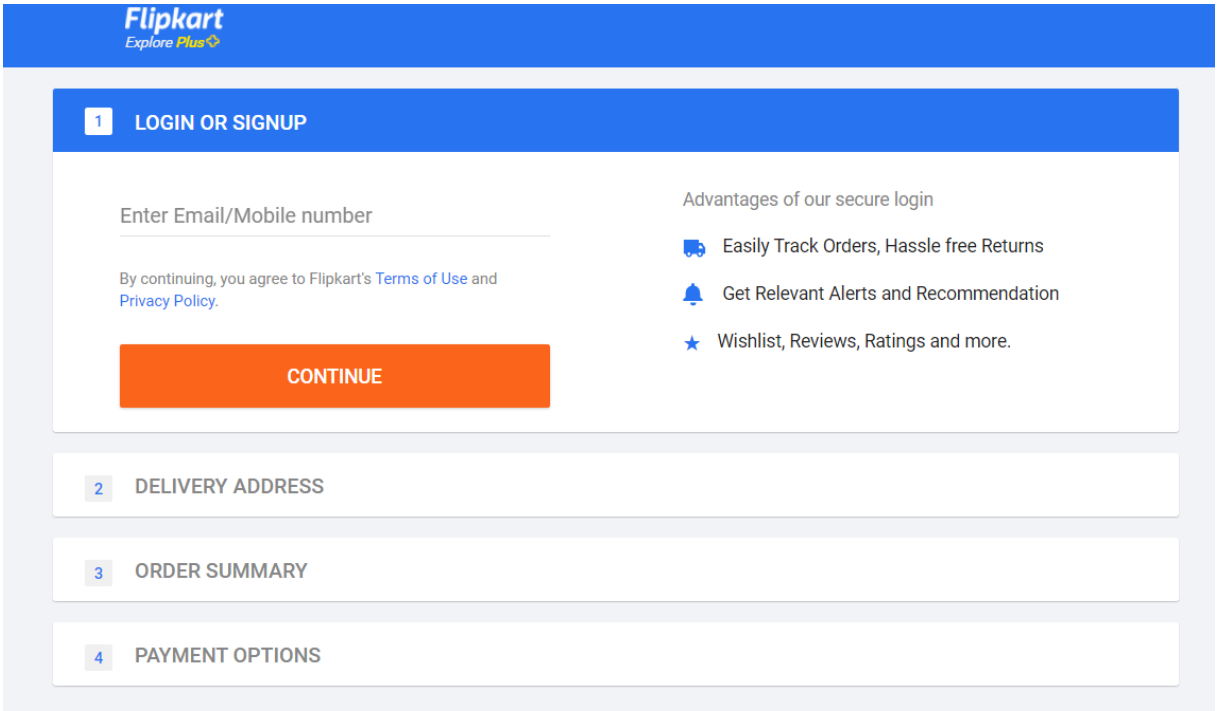


Figure 6.3.7 check out page

Check out page:

In this page we can add the shipping address, view the order summary and pay for the order.

CONCLUSION

Ultimately the project successfully solves the problem of difficult modes of purchasing products like going to electronics, clothes, furniture and grocery etc. It provides a easy user friendly application as well as website through which one with required internet connectivity can order products at any time in any moment at anywhere . It works on internet thus it can be used by anyone. A lots of research efforts from user side is reduced. User are provided with plenty of features such as ratings, reviews, verified sellers, door delivery, fast delivery, easy mode of transfer payment online and cancellation are embedded in app. We as team will always put efforts to improve the system as convenient to user.

REFERENCES

Abhijit Mitra 2013. “E-commerce in India- a review”, International Journal of Marketing, Financial Services and Management Research ISSN 2277- 3622 Vol.2,No. 2, February (2013)

Alka Raghunath, 2013. “Problem and Prospects of ECommerce “, International Journal of Research and Development - A Management Review (IJRDMR) ISSN (Print): 2319–5479, Volume-2, Issue – 1, 2013 68

Dr. Sachin Gupta, 2014. “Benefits and Drawbacks of MCommerce in India: A Review”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 4, April 2014.

E-Commerce Companies in India (n.d), http://comp aniesinindia.net/top-10-ecommerce-companies-in-india. html as viewed on 10.02.16

Electronic Commerce (n.d), http://www.investopedia.com/ terms/e/ecommerce.as viewed on 15.03.16

Gangeshwer, D. K. 2013.” E-Commerce or Internet Marketing: A Business Review from Indian Context”, International Journal of u- and e- Service, Science and Technology Vol.6, No.6.

Sarbapriya Ray 2011.” Emerging Trend of E-Commerce in India: Some Crucial Issues, Prospects and Challenges”, Computer Engineering and Intelligent Systems ISSN 2222- 1719 (Paper) ISSN 2222-2863 Vol 2, No.5, 2011

APPENDIX -A

RESEARCH PAPER

ISSN: 2230-9926

Available online at http://www.journalijdr.com

IJDR

International Journal of DEVELOPMENT RESEARCH

International Journal of Development Research
Vol. 6, Issue, 03, pp. 7253-7256, March, 2016

Full Length Research Article

A STUDY ON IMPACT OF E-COMMERCE ON INDIA's COMMERCE

[1,*]Dr. Rajasekar, S. and [2]Sweta Agarwal

[1]Principal, Akshaya Institute of Management Studies, Coimbatore – 641032, Tamilnadu, India
[2]Research Scholar, Bharathiyaar University, Coimbatore, Assistant Professor, Akshaya Institute of Management Studies, Coimbatore – 641032, Tamilnadu, India

ARTICLE INFO

Article History:
Received 22nd December, 2015
Received in revised form
24th January, 2016
Accepted 17th February, 2016
Published online 31st March, 2016

Key Words:

E-commerce,
E-finance,
E-Merchandise,
M- Commerce.

ABSTRACT

E-commerce involves an online transaction. E-commerce provides multiple benefits to the consumers in form of availability of goods at lower cost, wider choice and saves time. The general category of ecommerce can be broken down into two parts: E-Merchandise & E-finance. Many companies, organizations, and communities in India are doing business using E-commerce and also are adopting M-commerce for doing business. Ecommerce is showing tremendous business growth in India. Increasing internet users have added to its growth. Despite being the second largest user base in world, only behind China (650 million, 48% of population), the penetration of e-commerce is low compared to markets like the United States (266 M, 84%), or France (54 M, 81%), but is growing at an unprecedented rate, adding around 6 million new entrants every month. The industry consensus is that growth is at an inflection point. India's e-commerce market was worth about $3.9 billion in 2009, it went up to $12.6 billion in 2013. In 2013, the e-retail segment was worth US$2.3 billion. About 70% of India's e-commerce market is travel related. According to Google India, there were 35 million online shoppers in India in 2014 Q1 and is expected to cross 100 million mark by end of year 2016. By 2020, India is expected to generate $100 billion online retail revenue out of which $35 billion will apparel sales are set to grow four times in coming years. This paper is outcome of a review of various research studies carried out on Impact of E-commerce on Indian Commerce.

INTRODUCTION

India has emerged as one of the major players on the new international business scene. Its unstoppable economic growth since reforms in 1991 has become the focus of attention of researchers in the area of international business and management. The purpose of this paper is to review the impact of e-commerce on Indian Commerce that has been published in top business and management journals, with the aim of knowing what are the most influential papers, what are the issues that have received the most attention, which are the main findings or what more needs to be done in terms of research

E-COMMERCE

E-commerce is a paradigm shift. It is a ``disruptive'' innovation that is radically changing the traditional way of doing business.

****Corresponding author: Dr. Rajasekar, S.,***
Principal, Akshaya Institute of Management Studies, Coimbatore – 641032, Tamilnadu, India.

Electronic commerce is a type of business model, or segment of a larger business model, that enables a firm or individual to conduct business over an electronic network, typically the internet. E-commerce is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the Internet. These business transactions are business-to-business, business-to-consumer, consumer-to-consumer or consumer-to-business. The term *e-tail* is used in reference to transactional processes around online retail. E-commerce is conducted using a variety of applications, such as email, fax, online catalogs and shopping carts, Electronic Data Interchange (EDI), File Transfer Protocol, and Web services. It can be thought of as a more advanced form of mail-order purchasing through a catalog. E-Commerce is the movement of business onto the World Wide Web. The effects of e-commerce are already appearing in all areas of business, from customer service to new product design. It facilitates new types of information based business processes for reaching and interacting with customers like online advertising and marketing, online order taking and online customer service.

There has been a rise in the number of companies taking up e-commerce in the recent past. Major Indian portal sites have also shifted towards ecommerce instead of depending on advertising revenue. Many sites are now selling a diverse range of products and services from flowers, greeting cards, and movie tickets to groceries, electronic gadgets, and computers, etc

Historical Development of E-Commerce

A timeline for the development of e-commerce:

- 1971 or 1972: The ARPANET is used to arrange a cannabis sale between students at the Stanford Artificial Intelligence Laboratory and the Massachusetts Institute of Technology, later described as "the seminal act of e-commerce" in John Markoff's book *What the Dormouse Said*.
- 1979: Michael Aldrich demonstrates the first online shopping system.
- 1981: Thomson Holidays UK is first business-to-business online shopping system to be installed.
- 1982: Minitel was introduced nationwide in France by France Télécom and used for online ordering.
- 1983: California State Assembly holds first hearing on "electronic commerce" in Volcano, California. Testifying are CPUC, MCI Mail, Prodigy, CompuServe, Volcano Telephone, and Pacific Telesis. (Not permitted to testify is Quantum Technology, later to become AOL.)
- 1984: Gateshead SIS/Tesco is first B2C online shopping system and Mrs Snowball, 72, is the first online home shopper
- 1984: In April 1984, CompuServe launches the Electronic Mall in the USA and Canada. It is the first comprehensive electronic commerce service.
- 1990: Tim Berners-Lee writes the first web browser, WorldWideWeb, using a NeXT computer.
- 1992: Book Stacks Unlimited in Cleveland opens a commercial sales website (www.books.com) selling books online with credit card processing.
- 1993: Paget Press releases edition No. 3 of the first app store, The Electronic AppWrapper
- 1994: Netscape releases the Navigator browser in October under the code name Mozilla. Netscape 1.0 is introduced in late 1994 with SSL encryption that made transactions secure.
- 1994: Ipswitch IMail Server becomes the first software available online for sale and immediate download via a partnership between Ipswitch, Inc. and OpenMarket.
- 1994: "Ten Summoner's Tales" by Sting becomes the first secure online purchase.
- 1995: The US National Science Foundation lifts its former strict prohibition of commercial enterprise on the Internet.
- 1995: Thursday 27 April 1995, the purchase of a book by Paul Stanfield, Product Manager for CompuServe UK, from W H Smith's shop within CompuServe's UK Shopping Centre is the UK's first national online shopping service secure transaction. The shopping service at launch featured W H Smith, Tesco, Virgin Megastores/Our Price, Great Universal Stores (GUS), Interflora,Dixons Retail, Past Times, PC World (retailer) and Innovations.
- 1995: Jeff Bezos launches Amazon.com and the first commercial-free 24-hour, internet-only radio stations, Radio HK and NetRadio start broadcasting. eBay is founded by computer programmer Pierre Omidyar as AuctionWeb.
- 1996: IndiaMART B2B marketplace established in India.
- 1996: ECPlaza B2B marketplace established in Korea.
- 1998: Electronic postal stamps can be purchased and downloaded for printing from the Web.
- 1999: Alibaba Group is established in China. Business.com sold for US $7.5 million to eCompanies, which was purchased in 1997 for US $149,000. The peer-to-peer filesharing softwareNapster launches. ATG Stores launches to sell decorative items for the home online.
- 2000: The dot-com bust.
- 2001: Alibaba.com achieved profitability in December 2001.
- 2002: eBay acquires PayPal for $1.5 billion. Niche retail companies Wayfair and NetShops are founded with the concept of selling products through several targeted domains, rather than a central portal.
- 2003: Amazon.com posts first yearly profit.
- 2003: Bossgoo B2B marketplace established in China.
- 2004: DHgate.com, China's first online b2b transaction platform, is established, forcing other b2b sites to move away from the "yellow pages" model.
- 2007: Business.com acquired by R.H. Donnelley for $345 million.
- 2009: Zappos.com acquired by Amazon.com for $928 million. Retail Convergence, operator of private sale website RueLaLa.com, acquired by GSI Commerce for $180 million, plus up to $170 million in earn-out payments based on performance through 2012.
- 2010: Groupon reportedly rejects a $6 billion offer from Google. Instead, the group buying websites went ahead with an IPO on 4 November 2011. It was the largest IPO since Google.
- 2011: Quidsi.com, parent company of Diapers.com, acquired by Amazon.com for $500 million in cash plus $45 million in debt and other obligations. GSI Commerce, a company specializing in creating, developing and running online shopping sites for brick and mortar businesses, acquired by eBay for $2.4 billion.
- 2014: Overstock.com processes over $1 million in Bitcoin sales. India's e-commerce industry is estimated to have grown more than 30% from 2012 to $12.6 billion in 2013. US eCommerce and Online Retail sales projected to reach $294 billion, an increase of 12 percent over 2013 and 9% of all retail sales. Alibaba Group has the largest Initial public offeringever, worth $25 billion.
- 2015: Amazon.com accounts for more than half of all ecommerce growth, selling almost 500 Million SKU's in the US.

KEY DRIVERS IN INDIAN E-COMMERCE ARE

- Large percentage of population subscribed to broadband Internet, burgeoning 3G internet users, and a recent introduction of 4G across the country.

- Explosive growth of Smartphone users, soon to be world's second largest Smartphone user base.
- Rising standards of living as result of fast decline in poverty rate.
- Availability of much wider product range (including long tail and Direct Imports) compared to what is available at brick and mortar retailers.
- Competitive prices compared to brick and mortar retail driven by disintermediation and reduced inventory and real estate costs.
- Increased usage of online classified sites, with more consumer buying and selling second-hand goods
- Evolution of Million-Dollar startup like Jabong.com, Saavn, Makemytrip, Bookmyshow, Zomato Etc.

India's *retail market* is estimated at $470 billion in 2011 and is expected to grow to $675 billion by 2016 and $850 billion by 2020, – estimated CAGR of 10%. According to Forrester, the e-commerce market in India is set to grow the fastest within the Asia-Pacific Region at a CAGR of over 57% between 2012 –2016. India has an internet user base of about 354 million as of June of 2015. Despite being the second largest user base in world, only behind China (650 million, 48% of population), the penetration of e-commerce is low compared to markets like the United States (266 M, 84%), or France (54 M, 81%), but is growing at an unprecedented rate, adding around 6 million new entrants every month. The industry consensus is that growth is at an inflection point. In India, cash on delivery is the most preferred payment method, accumulating 75% of the e-retail activities. Demand for international consumer products (including long-tail items) is growing much faster than in-country supply from authorized distributors and e-commerce offerings. Largest e-commerce companies in India are Flipkart, Snapdeal, Amazon India, and Paytm.

Growth and Prospects of E-Commerce in India:

Increasing internet and mobile penetration, growing acceptability of online payments and favourable demographics has provided the e-commerce sector in India the unique opportunity to companies connect with their customers, it said. There would be over a five to seven fold increase in revenue generated through e-commerce as compared to last year with all branded apparel, accessories, jewellery, gifts, footwear are available at a cheaper rates and delivered at the doorstep, (as per industry body Assocham). It is noted that the buying trends during 2016 will witness a significant upward movement due to aggressive online discounts, rising fuel price and wider and abundant choice will hit the e-commerce industry in 2016.

It observed mobile commerce (m-commerce) is growing rapidly as a stable and secure supplement to the e-commerce industry. Shopping online through smart phones is proving to be a game changer, and industry leaders believe that m-commerce could contribute up to 70 per cent of their total revenues. In India roughly 60-65 per cent of the total e-commerce sales are being generated by mobile devices and tablets, increased by 50 per cent than in year 2015 and also likely to continue upwards. It noted that the browsing trends, which have broadly shifted from the desktop to mobile devices in India, online shopping is also expected to follow suit, as one out of three customers currently makes transactions through mobiles in tier-1 and tier-2 cities. In 2015, 78 per cent of shopping queries were made through mobile devices, compared to 46 per cent in 2013. In 2015, the highest growth rate was seen in the apparel segment almost 69.5 per cent over last year, followed by electronic items by 62 percent, baby care products at 53 per cent, beauty and personal care products at 52 per cent and home furnishings at 49 per cent. It revealed that Mumbai ranks first in online shopping followed by Delhi, Ahmedabad, Bangalore and Kolkata. On the mode of payment, almost 45 per cent of online shoppers reportedly preferred cash on delivery mode of payment over credit cards (16 per cent) and debit cards (21 per cent).

Only 10 per cent opted for internet banking and a scanty 7 per cent preferred cash cards, mobile wallets, and other such modes of payment, it said. Among the above age segments, 18-25 years of age group has been the fastest growing age segment online with user growth being contributed by both male and female segments. The survey revealed that 38 per cent of regular shoppers are in 18-25 age group, 52 per cent in 26-35, 8 per cent in 36-45 and 2 per cent in the age group of 45-60.

Challenges of E-commerce in India

India has less credit card population, lack of fast postal services in rural India. Accessing the Internet is currently hindered down by slow transmission speeds, frequent disconnects, cost of Wireless connection and wireless communication standards over which data is transmitted. High-speed-bandwidth Internet connection not available to most citizens of the nation at an affordable rate. In India, mostly people are not aware about the English language or not so good in English language. So that for the transaction over internet through electronic devices, language becomes one of the major factors to purchases, hire and sell a particular product or services. Multiple issues of trust in e-commerce technology and lack of widely accepted standards, lack of payment gateways, privacy of personal and business data connected over the Internet not assured security and confidentiality of data not in place to deploy ubiquitous IT Infrastructure and its maintenance

Conclusion

Growth of e-commerce depend to a great extent on effective IT security systems for which necessary technological and legal provisions need to be put in place and strengthened constantly. While many companies, organizations, and communities in India are beginning to take advantage of the potential of e-commerce, critical challenges remain to be overcome before e-commerce would become an asset for common people.

With the explosion of internet connectivity through mobile devices like Smartphone and tablets, millions of consumers are making decisions online and in this way enterprises can build the brand digitally and enhance productivity but government policies must ensure the cost effective methods/solutions. E-Commerce in India is destined to grow both in revenue and geographic reach. The challenge of establishing consumer trust in e-commerce poses problems and issues that need further research.

APPENDIX B- SRS

1. Introduction

1.1 Purpose:

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to consumers. Also, we shall predict and sort out how we hope this product will be used in order to gain a better understanding of the project, outline concepts that may be developed later, and document ideas that are being considered, but may be discarded as the product develops. In short, the purpose of this SRS document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how our client, team and audience see the product and its functionality. Nonetheless, it helps any designer and developer to assist in software delivery lifecycle (SDLC) processes.

1.2 Scope:

Primarily, the scope pertains to the E-Store product features for making online shopping. It focuses on the company, the stakeholders and applications, which allow for online sales, distribution and marketing of electronics. This SRS is also aimed at specifying requirements of software to be developed but it can also be applied to assist in the selection of in-house and commercial software products. The standard can be used to create software requirements specifications directly or can be used as a model for defining a organization or project specific standard. It does not identify any specific method, nomenclature or tool for preparing an SRS.

1.3 Definitions:

Configuration It means a product which is available / Selected from a catalogue can be customized.

CRM Customer Relationship Management

RAID 5 Redundant Array of Inexpensive Disk/Drives

SRS- Software Requirement Specification

GUI- Graphical User Interface

Stakeholder- The person who will participate in system

Ex. Customer, Administrator, Visitor etc.

1.4 References:

- The references are:

- Wikipedia.
- The Next Web.
- Flipkart Site(http://flipkart.com/about-us)
- YouTube

1.5 Overview:

Flipkart started with selling books. In 2010, they added to their catalogue media (including music, movies and games) and mobile phones and accessories. In 2011, product launches included cameras, computers, pens & office supplies, computer accessories, home and kitchen appliances, personal care, health care, gaming consoles, audio players and televisions. In 2012, product launches include health & beauty products, Life style products which includes watches, belts, bags & luggage. In November 2011, Flipkart launched a new Electronic Wallet feature that allows shoppers to purchase credit to their Flipkart account using credit or debit cards, and can subsequently be utilized to make purchases on the site, as and when required. From June 2012, Flipkart allowed people to buy toys, posters and from October 2012, Flipkart entered into apparel retailing.

2.Overall Description:

This document contains the problem statement that the current system is facing which is hampering the growth opportunities of the company. It further contains a list of the stakeholders and users of the proposed solution. It also illustrates the needs and wants of the stakeholders that were identified in the brainstorming exercise as part of the requirements workshop. It further lists and briefly describes the major features and a brief description of each of the proposed system. The following SRS contains the detail product perspective from different stakeholders. It provides the detail product functions of E-Store with user characteristics permitted constraints, assumptions and dependencies and requirements subsets.

2.1 Product Perspective:

This product aimed toward a person who don't want to visit the shop as he might don't get time for that or might not interested in visiting there and dealing with lot of formalities and work.

2.2 User Characteristics: User should be familiar with the terms like login, order system, payment etc.

2.3 Principle Actors: 2 Principle Actors are Web user and Administrator

2.4 General Constraints: A full internet connection is required for Flipkart.

2.5 Assumptions and Dependencies: Working of Flipkart need Internet Connection.

3. Specific Requirements:

3.1 Functional Requirements: This section provides requirement overview of the system. Various functional modules that can be implemented by the system will be -

3.1.1 Registration: If the web user wants to buy and analyze the product from the Flipkart, then he/she must be registered with his credentials, and an unregistered user can't login to the website and access the cart.

3.1.2 Login: the web user logins into the Flipkart portal by entering his/her valid user's name or id and password for the accessing the Flipkart online ecommerce website.

3.1.3 browse products: the web user uses the Flipkart portal or website to browse the products of his/her necessary requirements, there are many categories to choose from the catalog.

3.1.4 add items to cart: after viewing the required products the web user adds the necessary items to cart from which he can add or remove items as well as adjust the quantity of the products.

3.1.5 configure details: after adding the products to cart, the web user configures certain details such as delivery address, time and date of delivery, and any necessary remarks.

3.1.6 payment: in this system we deal with the payment and payment options. Such as mode of payment which include online or offline payment, if its online is it by credit or debit card, EMIs, and also by UPI.

3.1.7 checkout: the checkout page summarizes the details of the user and the ordered product.

3.2 Non-Functional Requirements:

Usability

The system shall provide a uniform look and feel between all the web pages. The system shall provide a digital image for each product in the product catalog.

Accessibility

The system shall provide multi language support.

Reliability & Availability

The system shall provide storage of all databases on redundant computers with automatic switchover. The system shall provide for replication of databases to off-site storage locations.

Internet Service Provider

The system shall provide a contractual agreement with an internet service provider for T3 access with 99.9999% availability.

Data Transfer

The system shall use secure sockets in all transactions that include any confidential customer information.

Data Storage

The customer's web browser shall never display a customer's password. It shall always be echoed with special characters representing typed characters.

3.3 Performance Requirements:

The product shall be based on web and has to be run from a web server. The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run. The performance shall depend upon hardware components of the client/customer.

3.4 Technical Issues:

This system will work on client-server architecture. It will require an internet server and which will be able to run PHP application. The system should support some commonly used browser such as IE, Mozilla Firefox, chrome etc.

4. Interface Requirement:

There are many types of interfaces as such supported by the E-Store software system namely; User Interface, Software Interface and Hardware Interface. The protocol used shall be HTTP. The Port number used will be 80. There shall be logical address of the system in IPv4 format.

Software Interface:

The user interface for the software shall be compatible to any browser such as Internet Explorer, Mozilla or Netscape Navigator by which user can access to the system. The user interface shall be implemented using any tool or software package like Java Applet, MS Front Page, EJB etc.

Hardware Interface:

Since the application must run over the internet, all the hardware shall require to connect internet will be hardware interface for the system. As for e.g., Modem, WAN – LAN, Ethernet Cross-Cable.

Communication Interfaces:

The flipkart e-store system shall use the HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol suite.

6. Software Design Description

6.1 Use-case diagram

6.2 DFD 6.1 Use-case diagram

USE CASE DIAGRAMS

Use case diagrams in unified modelling language helps us to identify and summaries the user's interaction with the system, it mainly contains actors and operations. In this Flipkart use case diagram show the scenarios of different actors interacting with the Flipkart system. Mainly three use case scenarios i.e,

USE CASE DIAGRAM 1:

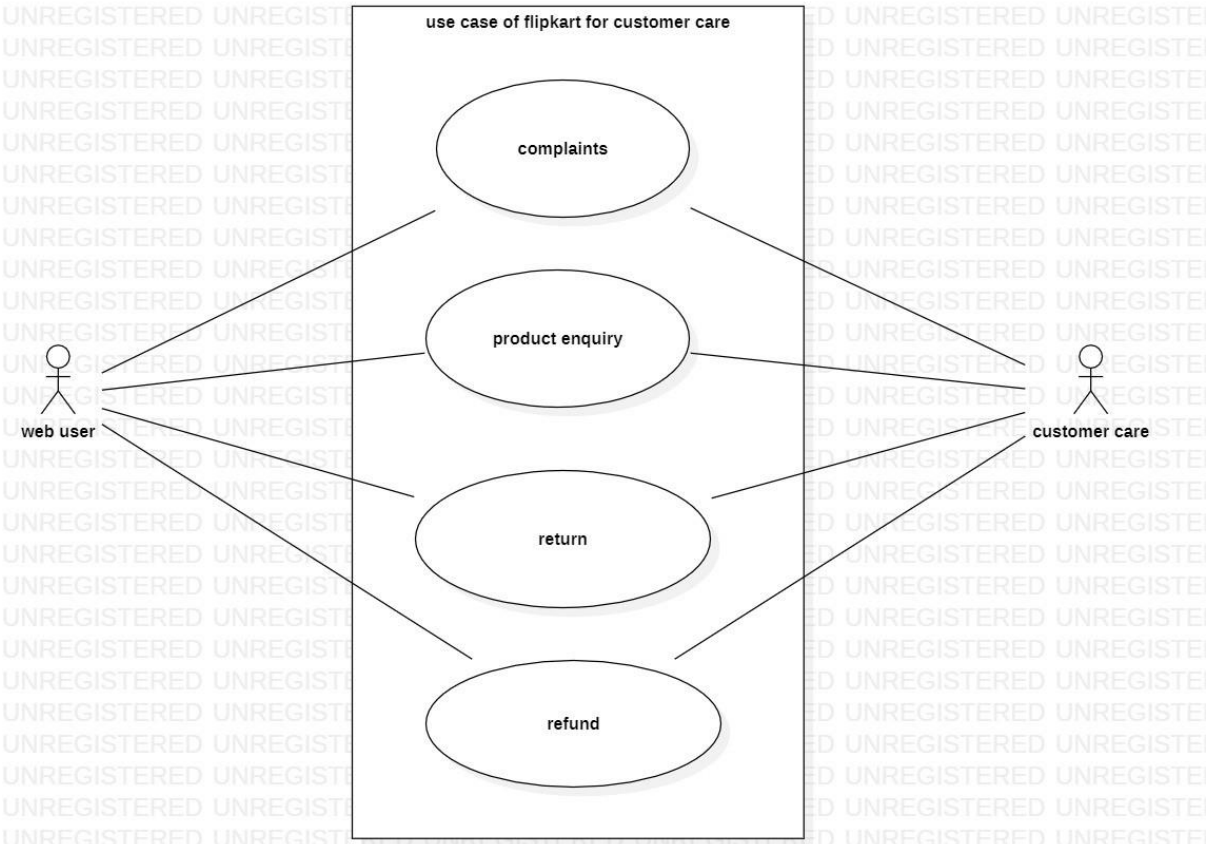


Figure 1: use case diagram for customer care

In this scenario the web user is using Flipkart customer care services such as the complaints regarding the products and orders, the required product enquiries, the time framed return and refund policy.

USE CASE DIAGRAM 2:

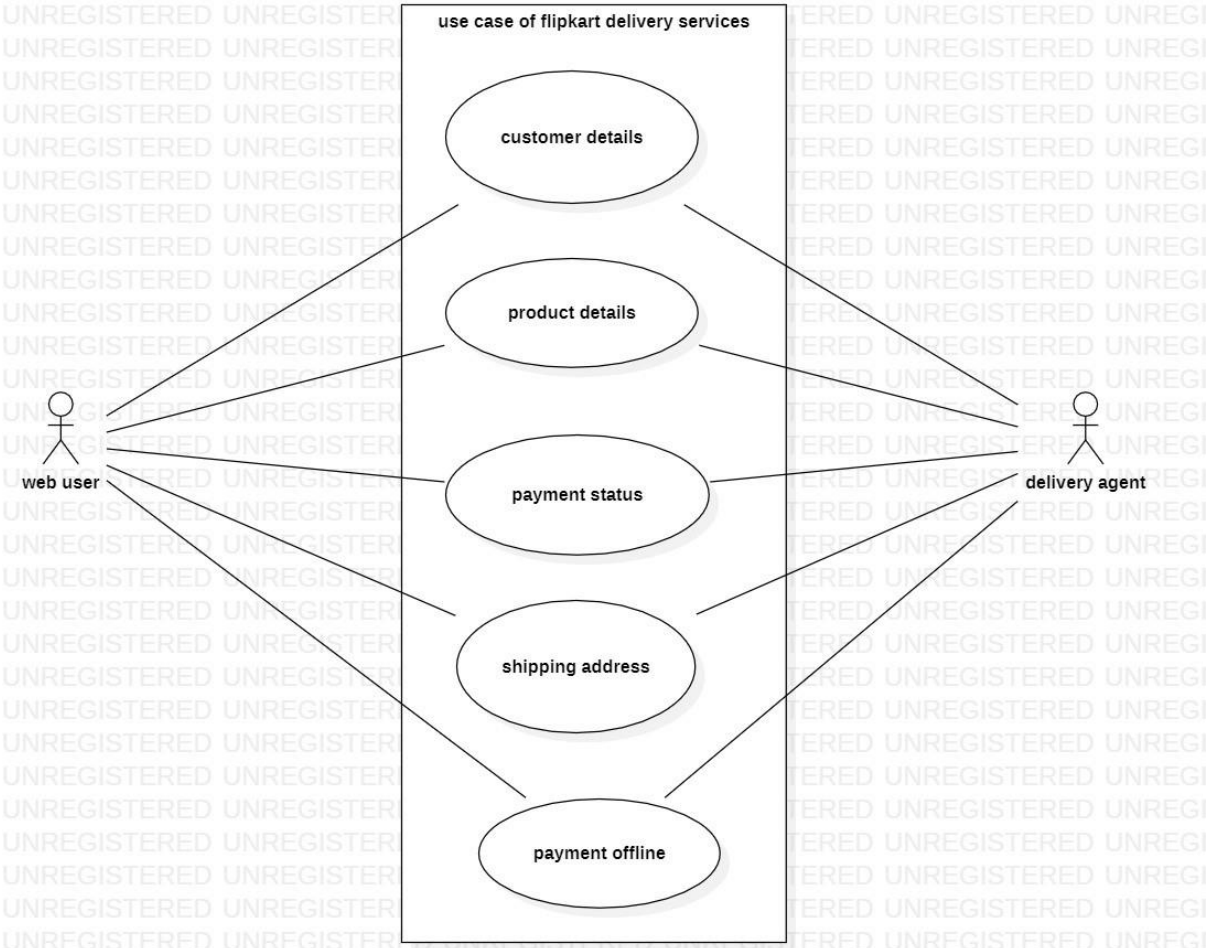


Figure 2: use case diagram for delivery

In this scenario the delivery agent is using the customer details, the product details, the payment status of the customer, the shipping address and also accepting the payment if it is done in offline mode.

USE CASE DIAGRAM 3:

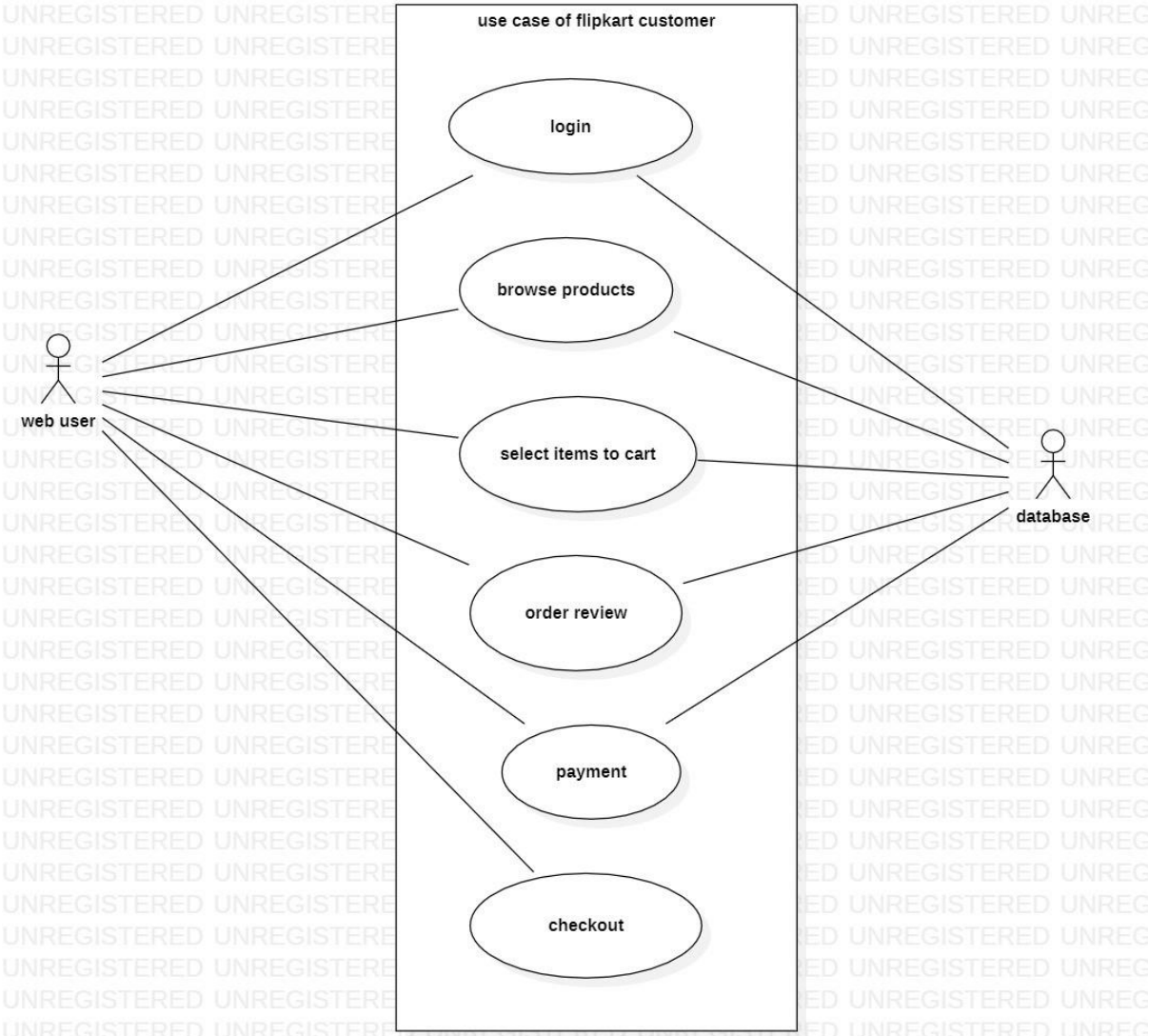


Figure 3: Use case diagram for Customer

In this scenario the customer or the web user uses flipkart portal for accessing the products, adding items to cart, verifying the order, and make payment for the suitable products required by the customer.

ZERO LEVEL DATA FLOW DIAGRAM

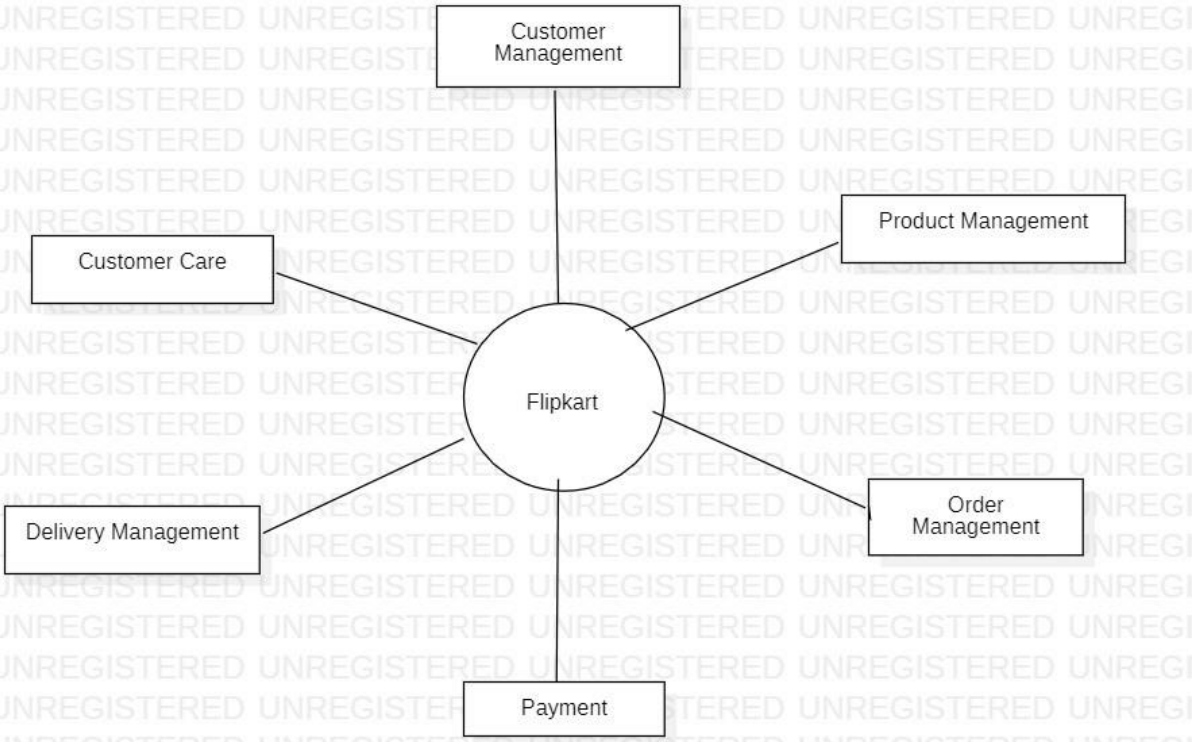


Figure 4: Data flow diagram level 0

The zero-level data flow diagram of unified modelling language for Flipkart, in which the basic overview of the flipkart ecommerce system is modelled it shows a glance landscape of single high-level processes with its relationship to its respective entities.

High level entities include:

b. Customer management
c. Product Management
d. Order Management
e. Payment
f. Delivery Management
g. Customer Care Management

LEVEL 1 DATA FLOW DIAGRAM:

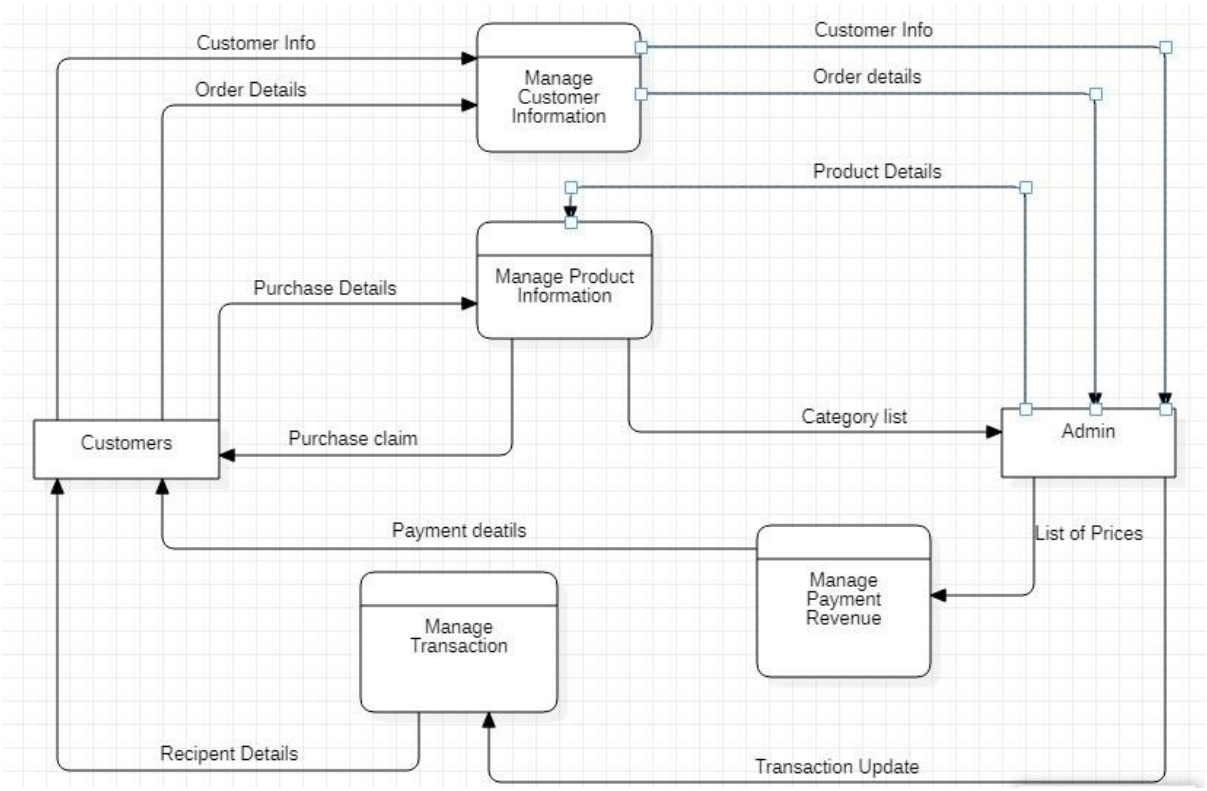


Figure 5: Data flow diagram level 1

The first -level data flow diagram of unified modelling language for Flipkart in which it represents the system which in turn is divided into system or sub processes where each process deals with one or more data flow transactions from its entities, as they hold provides the required functionality of flipkart.

ABBREVATIONS

App : Application

DFD : DATA FLOW DIAGRAM

DNS : Domain Name System

ER : Entity Relationship

GB : Giga Byte

IDE : Integrated development environment

INR : Indian Rupee

IOS : iPhone Operating System

J2EE : Java 2 Platform, Enterprise Edition

JSP : Java Server Pages

OTP : One-Time Password

QR : Quick Response

SSL : Secure Sockets Layer

UML : Unified Modeling Language

UPI : Unified Payments Interface

USD : United States Dollar

B2B : Business to Business

B2C : Business to Consumer

B2E : Business to Employee

CDN : Content Delivery Network

VPC : Virtual Private Clouds

PHP : Hyper Text Processor

CRM : Customer Relationship Management

TLS : Transport Layer Security

CTR : Click through Rate

PCI : Payment Card Industry

API : Application Programming Interface

CSR : Customer Service Representative

ASP : Average Selling Price

TPL : Third Party Logistics