Experiment No.: 1

Modeling using xADL

Learning Objective: Student should be able to do Modeling using xADL

Tool:- Apigen , Data Binding Library

Theory:

xADL is a highly-extensible software architecture description language (ADL). It is used to describe various aspects of the architecture of a software system. The architecture of a software system is its high-level design; design at the level of components, connectors, and their configurations. Like other ADLs such as Rapide, Darwin, and Wright, xADL's core models the four most common architectural constructs, namely:

- **Components** : (the loci of computation),
- **Connectors** : (the loci of communication),
- **Interfaces :** (the exposed entry and exit points for components and connectors), and
- **Configurations** : (topological arrangements of components and connectors as realized by **links**).

- xADL is an XML-based language. XML, the eXtensible Markup Language, was originally created to annotate, or "mark up" text documents with semantic information. Elements of text are marked up using tags, or special strings, that delimit a section of text. Tags begin with an open angle-bracket (<) and end with a closing angle-bracket (>). In XML documents, tags generally come in pairs, signifying the start and end of a text element. Start tags contain a tag name immediately after the opening angle-bracket, and end-tags contain the same name, prefaced by a forward slash (/) immediately after the opening angle-bracket. Elements may be nested as necessary, but may not overlap. An example of some marked up text in XML might be:

  <name><first>Herb</first> <last>Mahler</last></name>

- To HTML users, this format may look familiar. This is because XML and HTML both share a common historical ancestor, SGML (the Standard Generalized MarkupLanguage). In HTML, however, there is a finite set of allowed tags, each of which has a specific meaning dedicated to screen layout. So, tags like <H1> in HTML indicate that a text element is to be laid out in the Heading 1 style (usually large and bold, although this varies depending on the layout engine used), but do not indicate any other semantics

about the element--is it a story headline? Is it someone's name? This information is not present in HTML.

- This lack of a static set of allowed tags introduces a new problem into XML applications. What tags are allowed and what do they mean? How do two parties sharing marked-up documents come to an agreement on what elements are allowed, and where? How can they ensure that their information is marked up in a consistent way that is meaningful to both of them?
- The answer to this problem is to introduce a *meta-language*, a language for defining languages. In XML, meta-languages define what elements are allowed, where they are allowed to occur (and what their cardinality is), and what data may be part of each element. The XML 1.0 standard included such a meta-language, called the DTD (Document Type Definition) language. The DTD meta-language was sufficient for expressing XML-based languages as a set of production rules, much like a BNF (Backus-Naur Form) grammar, but proved insufficient for certain types of applications. To remedysome issues that users had with DTDs, the W3C (World Wide Web Committee) drafted anew meta-language for XML called XML Schema. XML schemas are more expressive than DTDs, they have an XML-like syntax (DTDs do not), and they allow type relationships among element types much like object-oriented inheritance.
- The development of XML schemas made it much easier for developers to create and evolve XML-based languages, and fostered the development of modular languages like xADL. A full treatment of XML is far out of scope for this guide.
- xADL's XML basis means that data in xADL is arranged hierarchically. Connections between data elements that are not hierarchically arranged are managed using simple XML links; xADL's tool support facilitates navigating these links.
- As an XML-based language, xADL documents are readable and writable by hand, as simple text documents. However, because xADL is defined in multiple schemas, each schema having its own XML namespace, the actual code can get quite complicated. For example, this is a real component description in xADL:

The xADL Type System

xADL adopts the more traditional types-and-instances model found in many programming languages. In this model, components, connectors, and interfaces all have types (called *component types*, *connector types*, and *interface types*, respectively).

Links do not have types because links do not have any architectural semantics. The relationships between types, structure, and instances are shown in the following table:

Instance (Run-time)	Structure (Design-time)	Type (Design-time)
Component Instance	Component	Component Type
Connector Instance	Connector	Connector Type
Interface Instance	Interface	Interface Type
Link Instance	Link	*(None)*
Group	Group	*(None)*

- Component Types
  - Sub architectures for Component Types
  - Signatures
- Connector Types*
  - Sub architectures for Component Types
  - Signatures
- Interface Types

Modeling of xADL

The instances schema gives xADL the ability to model running instances of architectural constructs like components, connectors, interfaces, and links. However, much work on software architecture is centered around the *design* of the architecture, rather than capturing properties of a running one.

For the purposes of this discussion, we make a distinction between architecture instances, which exist at run-time, and structural elements, which exist at design-time.

Run-Time	Design-Time
Instances	Structure

The xADL constructs available for modeling architectural structure mirror almost exactly those available for modeling architecture instances. The constructs defined in the instance schema are:

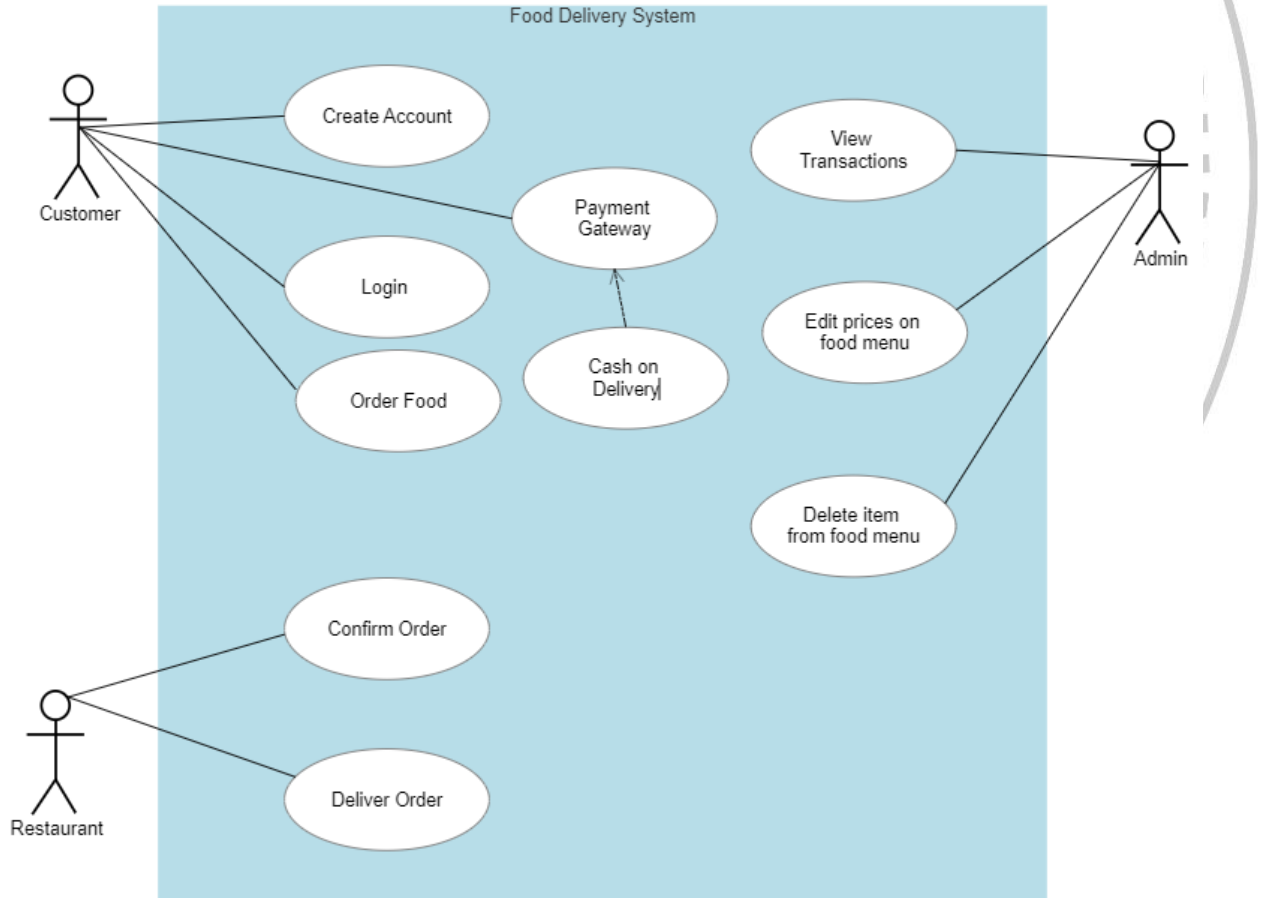
- Components
- Connectors

- Interfaces
- Links
- General Groups

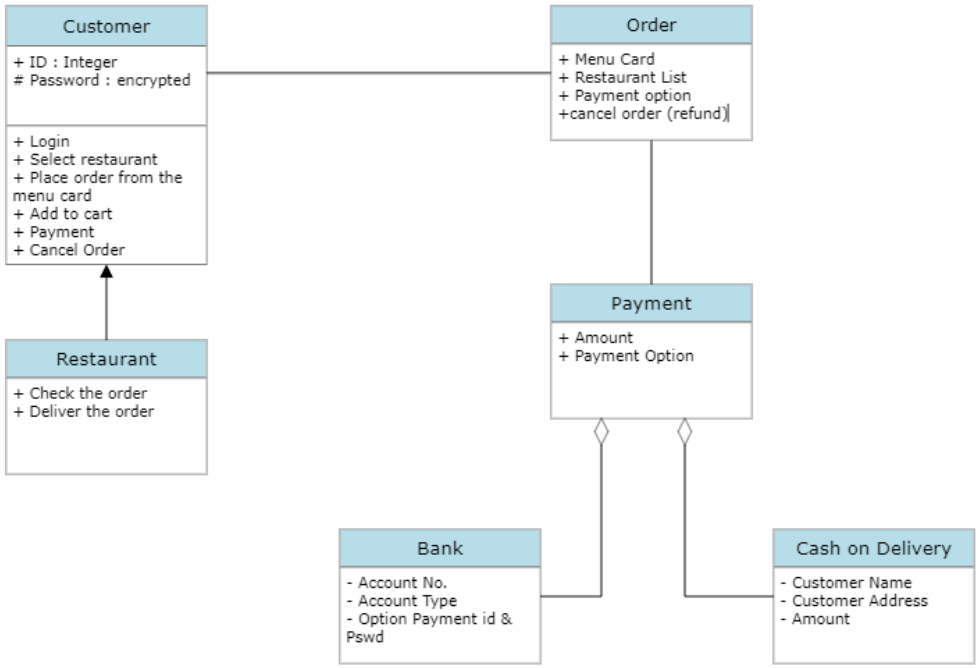
FORMAL DESCRIPTIVE LANGUAGE

- It should have login and registration features
- It should display menu and restaurant list
- Select restaurant from list and select food from menu
- Add to cart
- Customer selects payment option
- Payment options include cash on delivery, online transaction
- Check order and delivery order for restaurants
- Cancel order and refund

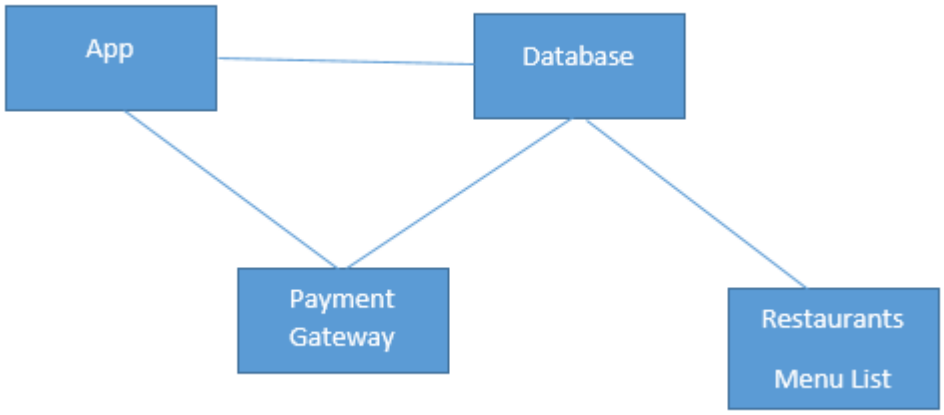
UML Diagram



Class Diagram



Box and Line:



"component" is used as a structural construct, as opposed to "component instance," which is used to describe a run-time instance. Similarly, "connector," "interface," and "link" are used instead of "connector instance, "interface instance," and "link instance.

Result and Discussion:

Learning Outcomes: Students should be able to

LO1: Define xADL.

LO2: Identify xADL Command.

LO3: Apply xADL Command for desired Output.

Course Outcomes: Upon completion of the course students will be able to do Modeling using xADL.

Conclusion:

Viva Questions:

1. Define xADL.
2. Explain xADL.
3. Explain syntax of xADL.
4. Explain modeling using xADL
5. Explain xADL type system.

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	