Data Modeling Assignment

1. List nouns that are candidate classes or attributes:

Classes	Attributes
Developer	userName
	password
	firstName
	lastName
	doB
	address
	phone
	email
	weblinks
Address	street1
	street2
	city
	state
	zipCode
Phone	countryCode
	areaCode
	phoneNumber
Widgets	id
	name
	topPosition
	bottomPosition
	rightPosition
	leftPosition
	width
	height
	color
	cssStyle
	cssClass
Website	websiteName
	title
	description
	createDate
	lastUpdate
	owner/developer
	visits
	links
Header Widget	shortText_size1
	shortText_size2
	shortText_size3
	shortText_size4
	shortText_size5
	shortText_size6

Paragraph Widget	longText
Image Widget	url
	topVerticalAlignment
	bottomVerticalAlignment
	middleVerticalAlignment
	leftHorizontalAlignment
	rightHorizontalAlignment
	centerHorizontalAlignment
Youtube Widget	videoID
	embeddable
	shareable
	expandable
Admin Page	

The above list represents various classes along with their respective attributes which can be inferred from the given data. As we can see there are many nouns that can be uniquely defined by a set of attributes. These unique nouns can be categorized as classes. These classes can be defined by a set of attributes, for example, Developer is a noun which can be considered as a class and can be defined by attributes such as user name, password, address, phone number, etc. Through the same way the above list was deduced.

2. List verbs as candidate relations between classes:

We have categorized various nouns as classes. Now we have to find relationship between these classes.

Classes	Relationship	Classes
Developers	CREATES	Website
Website	COMPOSED OF	Widgets
Widgets	COMPOSED OF	Header Widget
Widgets	COMPOSED OF	Paragraph Widget
Widgets	COMPOSED OF	Image Widget
Widgets	COMPOSED OF	Youtube Widget
Scripts	OPERATES ON	Widgets
Scripts	OPERATES ON	Websites

From the given data we can deduce the above mentioned relationships for the classes we have formed. These relationships creates a link between two classes. For example, Developers and Website are two different classes defined by their own attributes, from the given data we can see that a developer creates a website. So ‘CREATES’ becomes the relation between class Developer and Website. The same way other relationships can be deduced for other classes.

3. Generalization/specialization (inheritance):

   From the given data we can infer that there are three roles administrator, editor and reviewer. There is also the role of developer/owner who is the creator of webpage. All the 4 roles have few privileges in common, which are create, read, update and delete. Since these four privileges are common, it can be categorized into a class called Privileges and the other four roles will inherit these privileges from the class Privileges. The roles might have additional privileges which are applicable only to them.

4. Aggregation and/or Composition:

   We can come across few aggregation relations in the given data. For example, a Website can be composed of widgets, but a website can exist without widgets also. Hence aggregation exists between Website and Widgets. Similarly, Widget can be composed of Header or Paragraph or Image or Youtube Widget but not entirely of one type of widget (a widget might or might not be composed of entirely one type of widget, but not specified otherwise). Based on this we can infer that aggregation exists between these widgets and the main widget.

5. Classes vs Attributes analysis:

   As we can see we have come up with a list of classes and all the attributes that defines those classes. These classes are unique and have their own properties that define these classes. These properties that define the classes are known as attributes. For example to define a Developer we can use their name, address, date of birth, the websites they are associated with, etc. In the same manner a website's name, description, owner, etc. can define what the website is about and what is it about. If we compare all the attributes against its classes we can see that each attribute will define the class it belongs to in its own way.

6. Correct Data Types:

   Data types are similar to class, but in the case of data types its instance is identified by its value. Mentioning the correct data type is a very vital aspect of any class diagram. The data type specifies what kind of data is expected in the system. If a wrong data type is mentioned then providing data to that system is going to be useless since the system will be expecting different type of data. A class diagram might be considered as the first step in coming up with a whole system, so providing wrong data type at the initial step will make the system wrong all the way up to its end point.

7. Cardinality - number of instances participating in a relation:

   If we look at the given data we can see the number of instances participating in a relation. It specifies the uniqueness of data value. For example, it is mentioned that an address can be marked as primary address, which means that at least one address should be present to define the Developer class. Also it states that additionally an address can also be marked as billing address, so an address there can exist multiple addresses among which one can be primary and one more can be billing address. Therefore its cardinality 1...* which specifies at least one address should exist all time. The same applies to phone number. Similarly we can find other cardinalities from the given data.

8. Reify:

   There are few privileges common for all the four roles mentioned in the given data. So rather than writing all the privileges to each class of roles we can create a class called Privileges and include the common privileges in it. All the roles can inherit these privileges from this class.

   There are scripts that particularly operate on Pages, also few script that operate on widgets. These scripts can be written in its respective classes where it is applicable. But there are also other scripts which are common to both pages and widgets. These common scripts can be written in a separate class and can be associated to both the classes.

   It is the same case with widgets. There are few common attributes and some particular attributes that are applicable to youtube widget or paragraph widget or image widget or header widget individually. These common attributes can be written in class Widget and individual attributes can be written separately in its own class.

   Prose:

   Based on the given set of data I have come up with the above mentioned information. The data can be classified into various classed such as Developer, Privileges, Widgets, etc. These classes can be defined by a set of attributes. Attributes such as name of the developer, date of birth, address, etc. These attributes are unique to its classes and gives a brief description about what the class is about. Classes may also be associated with methods, for example, class Developer has methods such as assign privilege, revoke role, assign role, etc. These methods allows the class to perform various activities with other classes or entities with which a relationship exists.

   Relationship defines what kind of link exist between two classes. Between the class Developer and Website there exists a relationship CREATES, this specifies that a

developer can create a website. So basically a relationship tells us what kind of activity one class can perform on the other.

According to the given data there are four different roles. There are four privileges which are common to these roles. Also there are additional privileges for certain roles. So instead of writing these common privileges to all the roles we can create a class called Privileges and it can hold all the common privileges, all the four roles can inherit these common privileges from the class Privileges. This way we can eliminate redundancy in the data.

Aggregation and composition in terms of UML class diagram states the dependencies between two entities. Aggregation is where the child can exist without the parent and composition is where the child cannot exist without the parent. We can see in the given data there exists an aggregation between widgets and header/paragraph/youtube/image widget. Widgets are made up of other kind of widgets, but not necessarily that it should me made of only widgets or only one kind of widgets(since not mentioned in the given data). This is a good example for aggregation.

A class has a set of unique attributes. These unique attributes are the defining property of the class, it defines the class. For example, class Developer has attributes such as name, address, phone number, date of birth, etc. These details are unique to this class and it helps in defining the class Developer.

Cardinality helps to identify the number of instances participating in a particular relationship. It shows how unique the value is. It is mentioned in the data that there exists a primary phone number, so it implies that there always exists a phone number i.e. at least one value of phone always exists, hence the cardinality 1…*. Whereas it is not mentioned that for a given widget header widget is a must, it might or might not contain header widgets, hence the cardinality 0…*.

There is a lot of redundancy in the given data, such as the activities of a developer, details a developer needs to provide to create a website, roles and privileges of a developer. All these data can be abstracted and reduced to minimal.

Assumptions:

- There always exists a developer.
- Address and phone number should be provided to create a developer account.
- If a developer account is created, there is no compulsion that a website/webpage should exist or be associated with that developer. Developer may or may not create a webpage.
- A webpage may or may not contain widgets.
- Widgets might or might not be made up of header/paragraph/image/youtube widgets

- Roles such as editor, administrator and reviewer might be present or might not be present.
- There might or might not be any scripts that operate on webpages or widgets.