**UML (Unified Modeling Language):**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

So UML can be described as a general purpose visual modeling language to visualize, specify, construct and document software system. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc.

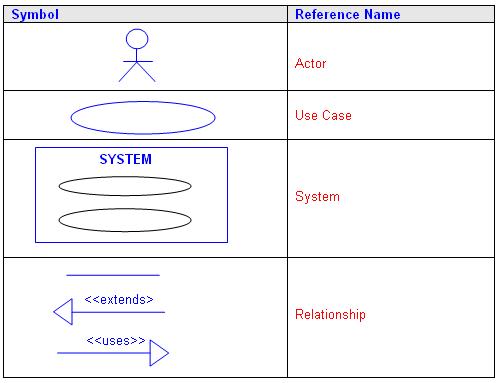
**Use case diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.Use case diagrams are a set of use cases, actors and their relationships. They represent the use case view of a system.

The purposes of use case diagrams can be as follows:

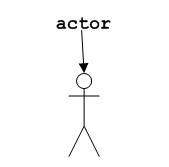
* Used to gather requirements of a system.
* Used to get an outside view of a system.
* Identify external and internal factors influencing the system.
* Show the interacting among the requirements are actors.

**UML Use Case Diagram Symbols:**



**Actor:**

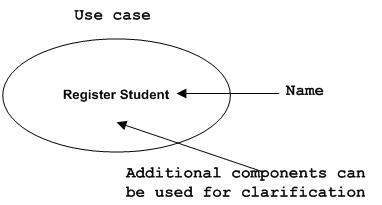
An actor can be defined as some internal or external entity that interacts with the system.



Actor is used in a use case diagram to describe the internal or external entities.An actor is something or someone that interacts with the system.Actor communicates with the system by sending and receiving messages.An actor provides the stimulus to activate an Use-case.Message sent by an actor may result in more messages to actors and to Use-cases.Actors can be ranked: primary and secondary; passive and active.Actor is a role not an individual instance.

### Use case :

Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.



Use case is used to capture high level functionalities of a system.For each actor ask these questions:

* The actor require the functions from the system?
* Functionalities of actor.
* The actor’s work could be simplified or made efficient by new functions in the system?
* The problems with the existing systems?
* The inputs and outputs of the system?

**<<include>> Relationship:**

* A connection between two use cases
* Indicates a use case that is used (invoked) by another use case
* Links to general purpose functions, used by many other use cases

**<<extend>> Relationship:**

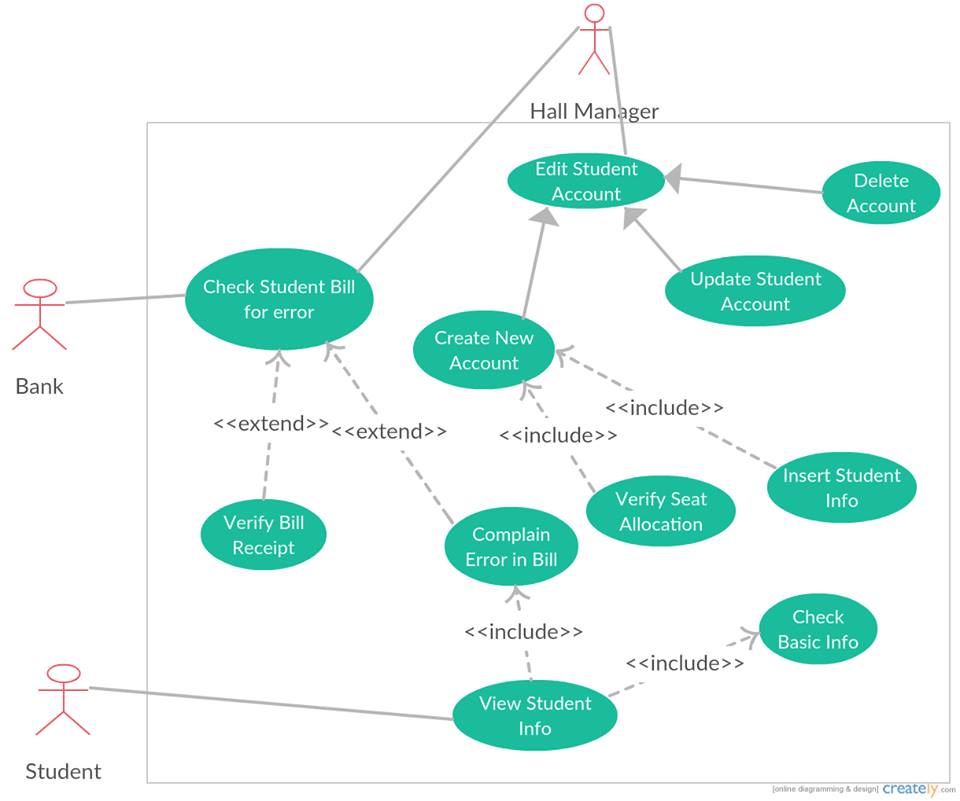
* A connection between two use cases
* Extends a use case by adding new behavior or actions
* Specialized use case extends the general use case

**Use Case Scenario:**

A use case collects scenarios. By writing out your main success story in a textual format, you’ll have a clearer outlook on the scope of your project. Before you ever start diagramming, it’s a good idea to build your use case as a series of written bullet points. A use case is a collection of related success and failure scenarios that describe actors using a system to support a goal

There are three types of use case diagrams used in our projects:

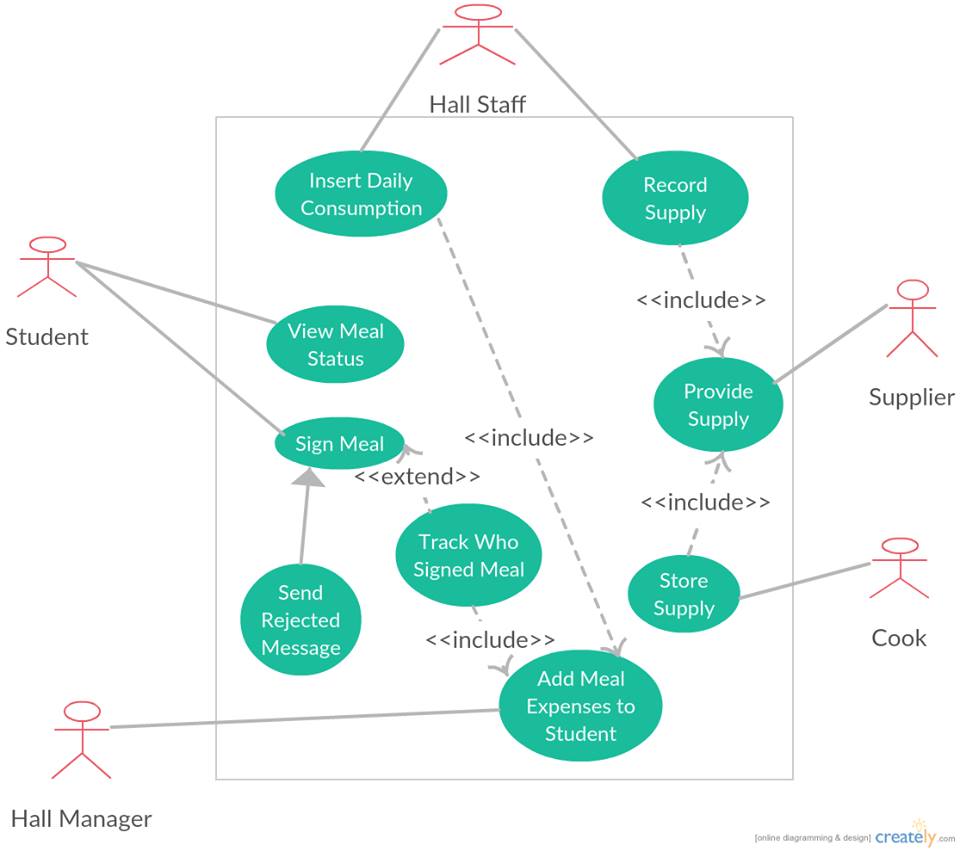
1.UCD on student account control:



Description:

Here student acts as an actor for the use case diagram. He checks student bill through check student bill use case. This use case extends to verify and complain bill. Hall manager acts as an actor for edit student account which includes create account also. Verify seat and insert student info are include use cases of create student account. complain error ‘includes’ view student info where student is an actor.

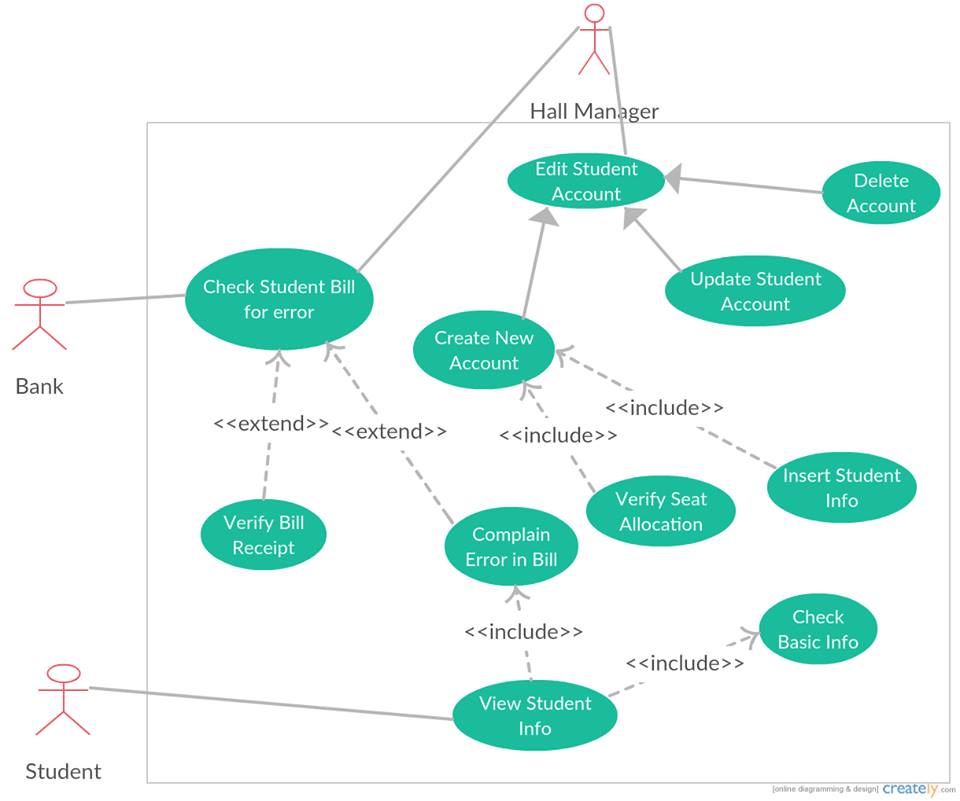
2.UCD ON student meal sign:



Description:

4 actors are here. At first student acting for meal sign use case. This can be rejected by reject message process. Hall stuff acts for insert and record use cases. Insert includes add meal expense use case as well where hall manager is an actor. And record use case’s include use case is supply so here supplier is an actor. Cook also is an actor acts for store supply use case where it ‘include’ provide supply use case.

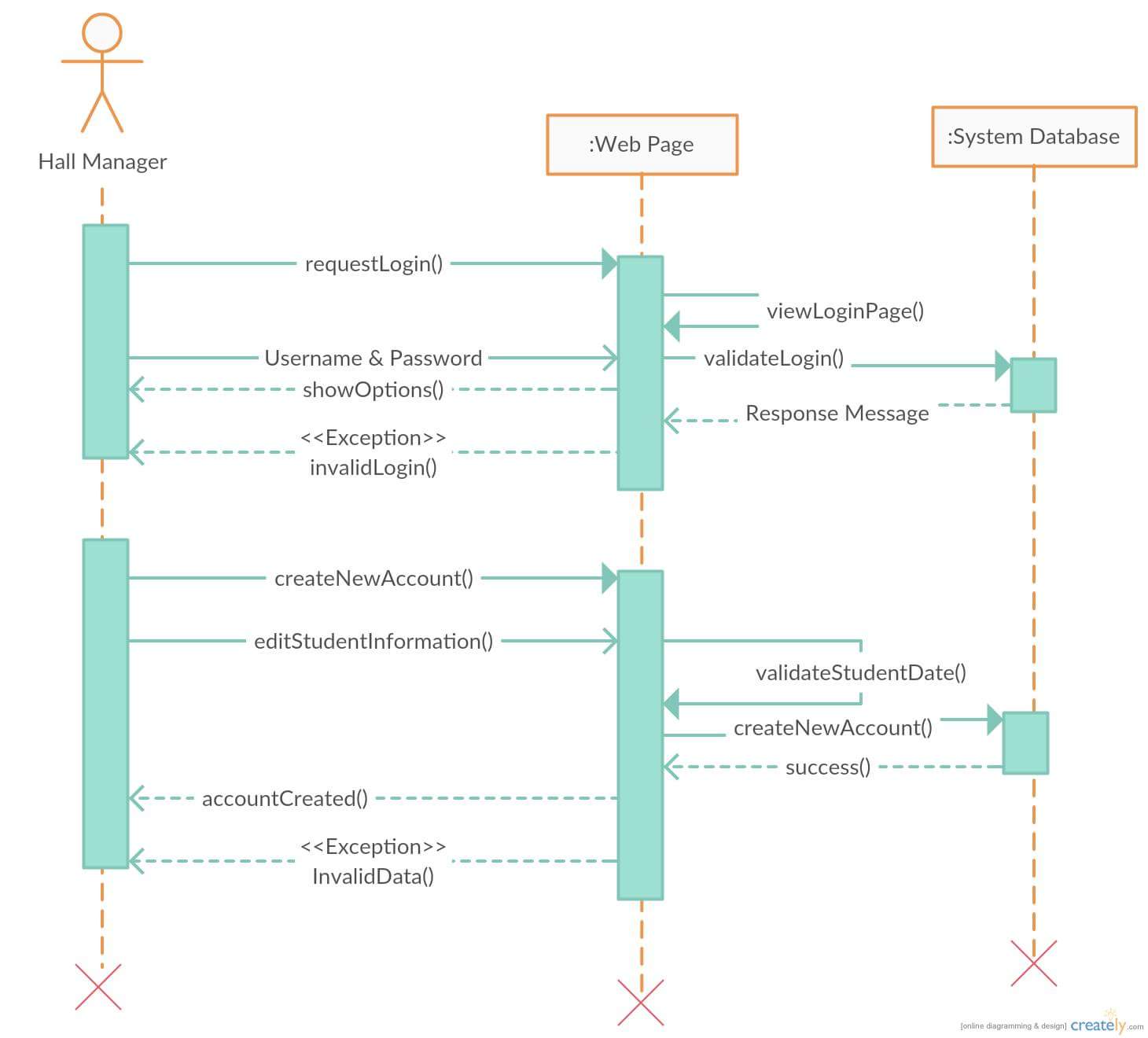
3.UCD On student view bill:



Description:

Student acts as an actor who is out of the boundary. He can pay bill and view bill. Those are use cases. Pay bill introduces two more use cases collect bill book and submit bill receipt so they are include version of pay bill. Bank works as another actor here for paying bill. And variation to this use case is submit fine. So this the extended version. Bill are of hall bill and mess bill these two types which initiates the use case insert bill in include section. Hall stuff acts as an actor for inserting bill and creating bill receipt to update the database. Whereas Hall manager acts on collecting bill receipt use case.

* 1. **Sequence diagram for student’s account:**

****

**Fig:** Sequence Diagram for student’s account

**1.Use case scenario for student account:**

**Use case title:** Osmany\_Hall\_Messing\_System\_student\_account UC1

**Use case description:** This use case describes the flow of managing student accounts

**Primary actor:** Hall manager

**Stakeholders and interests:**

-Students: who will sign meal and view bills.

-hall manager: who will open student accounts and update necessary informations .

**Preconditions:**

1.Student must be hall resident.

2. Student must open an account:

**Post conditions:**

User will be able to sign meal and view bills successfully.

**Main success scenario:**

1. Hall manager will open a student account

2. He will provide the account name and password to the student.

3. Students can access their account after that.

4. They can manage their profiles.

5. They can add their photos.

6. They can also modify their necessary informations.

**Alternative flows:** N/A

**Special Requirements :**

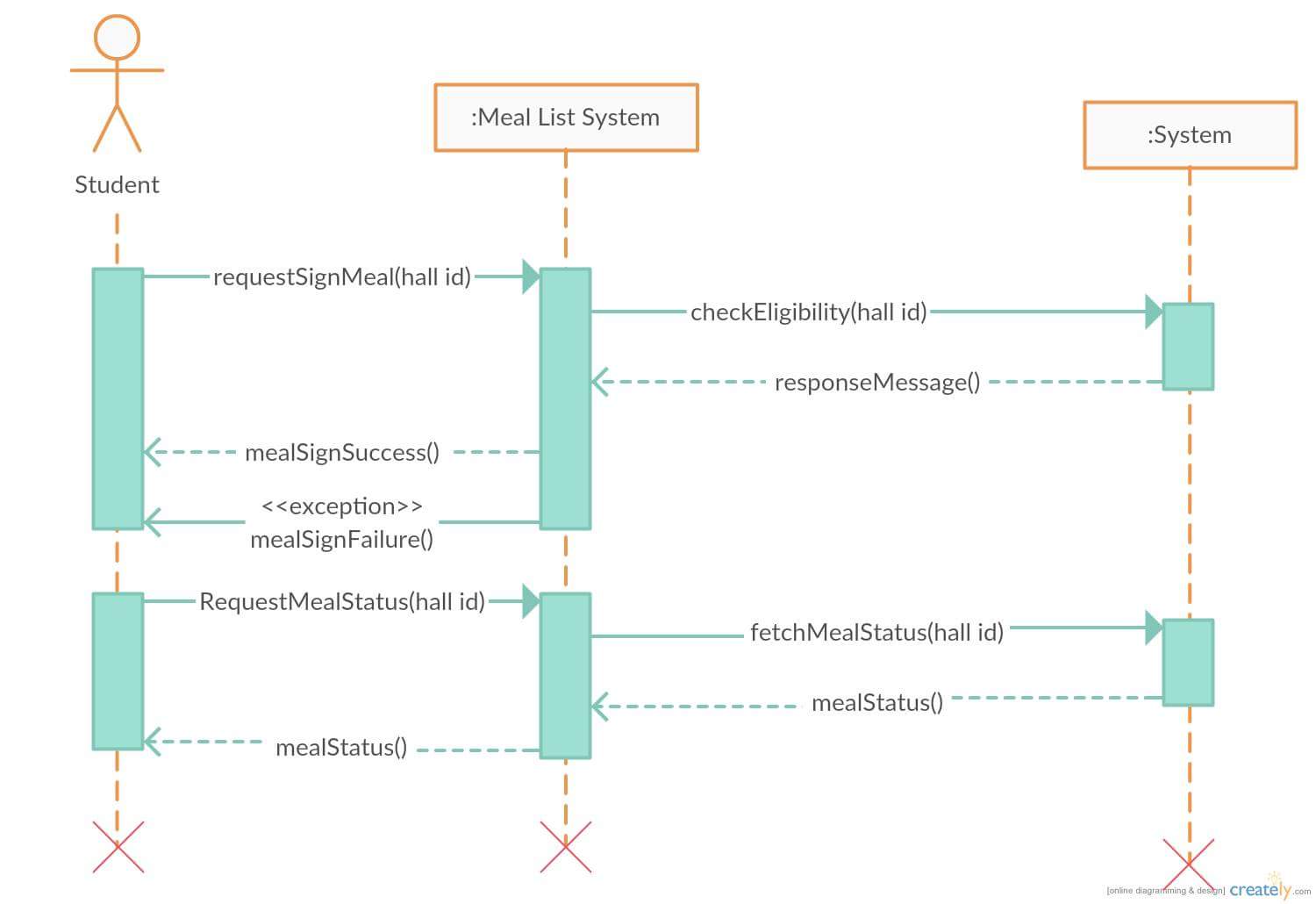
* Software must be user friendly and secured.
* Compatible devices must be pre determined.
* Database management must ensured.
* Invalid login must be prevented.

**Technology and Data variations list :**

* Validity of data entered by user must be checked.
* Proper data format must be mentioned.
* Data compatibility must be must be maintained.

**Frequency of operations :** Could be nearly continuous.

* 1. **Sequence diagram for signing meal:**



**Fig:** Sequence diagram for signing meal

**2.Use Case Scenario for signing meal:**

**Use case UC1:**Osmanyhall\_messing\_system UC2

**Use case description:** This use case describes the flow of signing meal of a hall student

**Primary Actor:** Student

**Stakeholders and Interests:**

* **Student:** Wants to order meals of next day perfectly without any error .
* **Hall Cook:** Wants to get total number of orders group by meal-time.
* **Hall Management :** want to get proper track of meal, it’s count and student and grand totals for each individuals.
* **Hall Manager :** Wants to get monthly bill without errors perfectly and categorized.

**Pre Condition :** Student is identified and unique.

**Post Condition :** Order is saved and total number of order is calculated as per meal time and also as per student.

**Main Success Scenario :**

1. Student log in to his account and order for meal.
2. Meal ordered as per student and also as per meal time .
3. Meal ordered for next day and cook get total count of meals as per meal-time.
4. Hall management gets total number of meal for individual students and for given time duration.
5. Each student gets his/her monthly bill from the system with proper report of meals
6. Students pays bill and updates it to their account.
7. System shows each students payment reports and also meal reports perfectly.

**Extensions :**

1. If system fails :
   1. Hall management ensures restarting system.
   2. Students to check their account and meal states.
2. Orders should be made before starting calculation for next day.
3. Time limit for ordering must be maintained.
4. User validation must be ensured.
5. Cancelation of orders must be in time limit.
6. Payments must be updated in time properly.

**Special Requirements :**

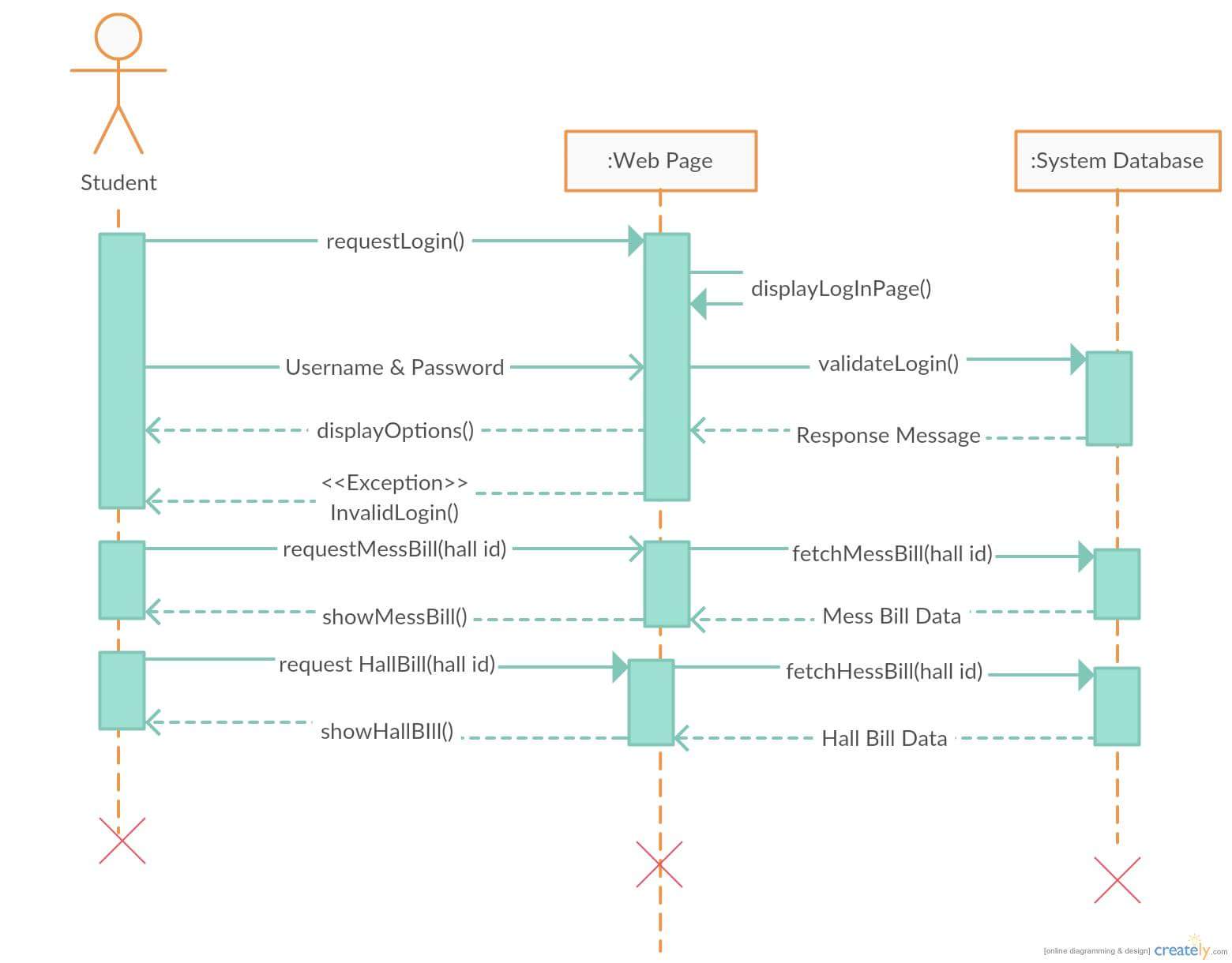
* Software must be user friendly and secured.
* Compatible devices must be pre determined.
* Database management must ensured.
* Invalid login must be prevented.

**Technology and Data variations list :**

* Validity of data entered by user must be checked.
* Proper data format must be mentioned.
* Data compatibility must be must be maintained.

**Frequency of operations :** Could be nearly continuous.

* 1. **Sequence diagram for view bill:**



**Fig:** Sequence diagram for view bill

**3.Use Case Scenario for view bill:**

**Use case title:** Osmanyhall\_messing\_system\_view\_bill uc3

**Use case description:** This use case describes the flow for viewing the bill of a hall student

**Primary Actor:** Student

**Stakeholders and interests:**

* Student-who can view their mess and meal bill.

**Pre conditions:**

1. Student must be an authorized user.

2. Student must have view bill privilege in the system.

3. The bill information must be presented in the system.

**Post conditions:**

Student must be able to see his mess bill and hall bill.

**Main success scenario/Normal flow:**

1. Student: initiates to view meal information.

2. System: will give students two options, hall bill and mess bill.

3. Student: will choose one of the options.

4. Student: will display the bill based on selected options.

5. Students: bill information in database is updated by administrator.

6. Student: can see their individual bills for months.

**Alternative flow:** N/A

**Special Requirements :**

Software must be user friendly and secured.

Compatible devices must be pre determined.

Database management must ensured.

Invalid login must be prevented.

**Technology and Data variations list :**

Validity of data entered by user must be checked.

Proper data format must be mentioned.

Data compatibility must be must be maintained.

**Frequency of operations :** Could be nearly continuous.

**Sequence diagram:**

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are sometimes called event diagrams or event scenarios.

**The purposes of sequence diagram:**

* A sequence diagram is a good way to visualize and validate various runtime scenarios.
* These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.
* Often used to show the processing described in use case scenarios.
* Used to show the overall pattern of the activities or interactions in a use case.

**System Sequence Diagram**   
Notations:

|  |  |
| --- | --- |
| **Actor** |  |
| **System** |  |
| **Lifelines** |  |
| **Activation bars** |  |
| **Input messages** |  |
| **Output messages** |  |
| **Frame** |  |

Description:

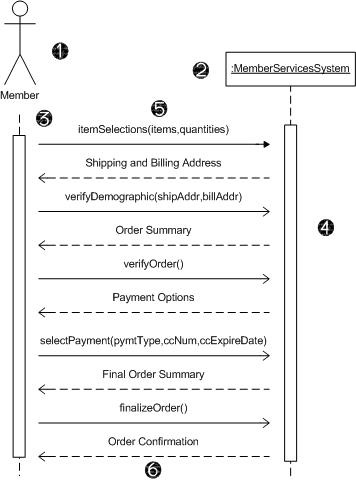
* **Actor** - the initiating actor of the use case is shown with the use case actor symbol.
* **System** – the box indicates the system as a "black box" or as a whole. The colon (:) is standard sequence diagram notation to indicate a running "instance" of the system.

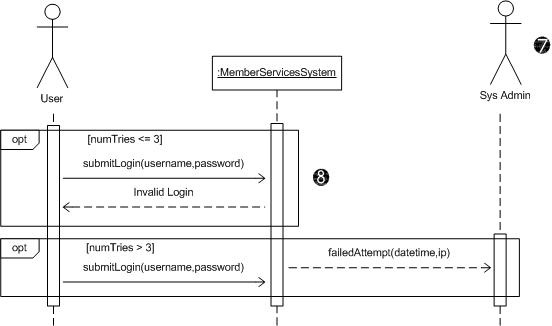
**Lifelines** – the dashed vertical lines extending downward from the actor and system symbols, which indicate the life of the sequence.

**Activation bars** – the bars set over the lifelines indicate period of time when participant is active in the interaction.

**Input messages** - horizontal arrows from actor to system indicate the message inputs. UML convention for messages is to begin the first word with a lowercase letter and add additional words with initial uppercase letter and no space. In parentheses include parameters, following same naming convention and separated with commas.

* **Output messages –** horizontal arrows from system to actor shown as dashed lines. Since they are web forms, reports, e-mails, etc. these messages do not need to use the standard notation.
* **Receiver Actor**:other actors or external systems that receive messages from the system can be included.
* **Frame** : A box can enclose one or more messages to divide off a fragment of the sequence. These can show loops, alternate fragments, or optional steps. For an optional fragment the condition shown in square brackets indicates the conditions under which the steps will be performed.





Steps of using sequence diagram:

* 1.Modeling and documenting how the system will behave in various scenarios.
* 2.Validating the logic of complex operations and functions.

UML class diagrams are useful when modeling business data. By accurately modeling attributes and associations of class entities, we can easily map these class diagram specifications to entity beans with CMP. Class attributes map to abstract access methods for persistent fields, and association roles map to abstract access methods for relationship fields. Navigability determines whether relationship access methods appear in both related entity beans or just one. Furthermore, multiplicity notation determines the correct type for relationship fields, life cycle issues, and cascading delete characteristics.