**Introduction:**

The Unified Modeling Language (UML) is a general-purpose, developmental, [modeling language](https://en.wikipedia.org/wiki/Modeling_language) in the field of [software engineering](https://en.wikipedia.org/wiki/Software_engineering), that is intended to provide a standard way to visualize the design of a system.

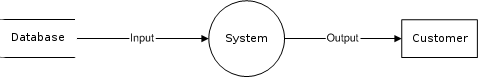
Data Flow Diagram:

Data flow diagram (DFD) projects an overview of an information system through representing the production and receive of 'data'. It is a graphical representation of the "flow" of data through an information system, modelling its *process* aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the [visualization](https://en.wikipedia.org/wiki/Data_visualization) of [data processing](https://en.wikipedia.org/wiki/Data_processing) (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a [flowchart](https://en.wikipedia.org/wiki/Flowchart)).

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

We usually begin with drawing a context diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with additional information about the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and so on is possible but anything beyond level 3 is not very common.



Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method [SSADM](https://en.wikipedia.org/wiki/SSADM). The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report. How any system is developed can be determined through a data flow diagram model.

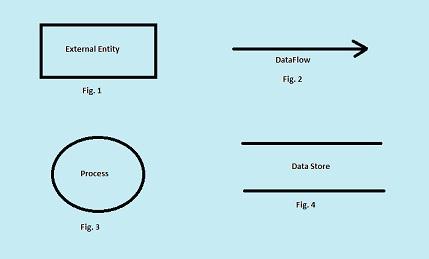
In the course of developing a set of *levelled* data flow diagrams the analyst/designer is forced to address how the system may be decomposed into component sub-systems, and to identify the [transaction data](https://en.wikipedia.org/wiki/Transaction_data) in the [data model](https://en.wikipedia.org/wiki/Data_model).

Data flow diagrams can be used in both Analysis and Design phase of the [SDLC](https://en.wikipedia.org/wiki/Systems_development_life_cycle).

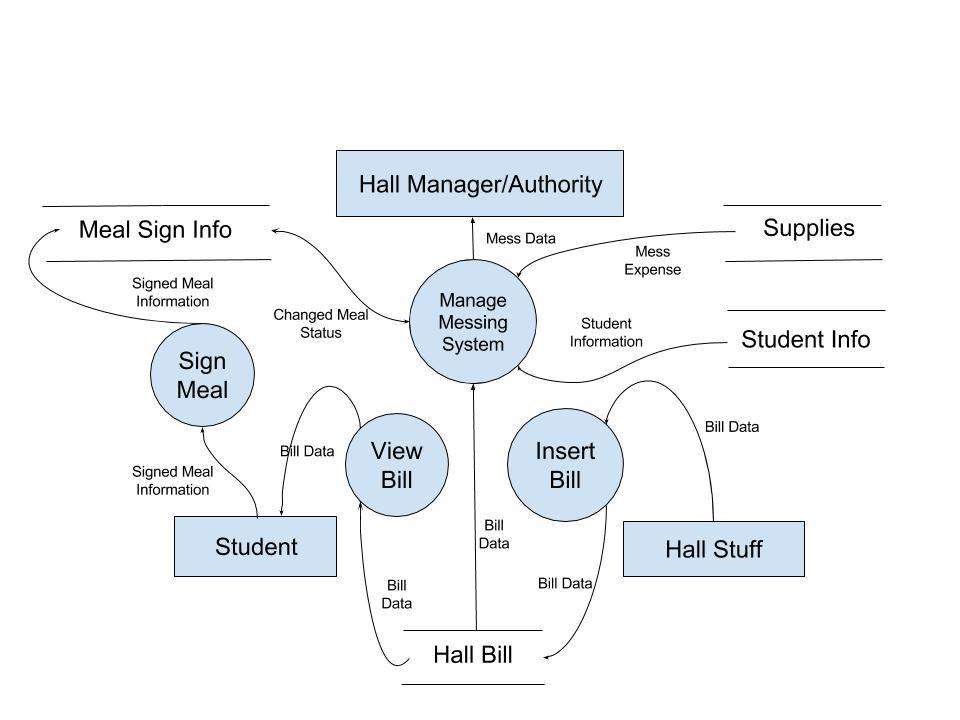
There are different notations to draw data flow diagrams defining different visual representations for processes, data stores, data flow, and external entities.

Using any convention’s DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

1. **External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
2. **Process:**any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.
3. **Data store:** files or repositories that hold information for later use, such as a database table or a membership form.
4. **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the



Osmani hall messing system data flow diagrams:

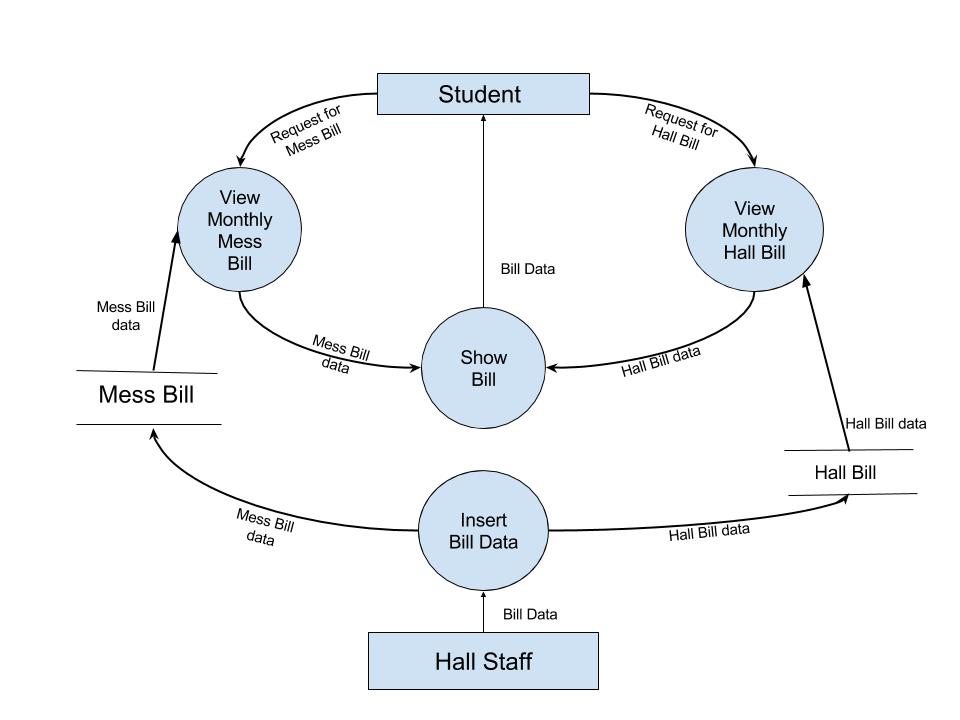
our whole project can be summarized within the level 0 dfd below:

The data flow diagram of our project was designed according to our projects features. The main 4 features were used as criteria to create four different data flow diagrams. Those are given below:

1. Check bill for students:

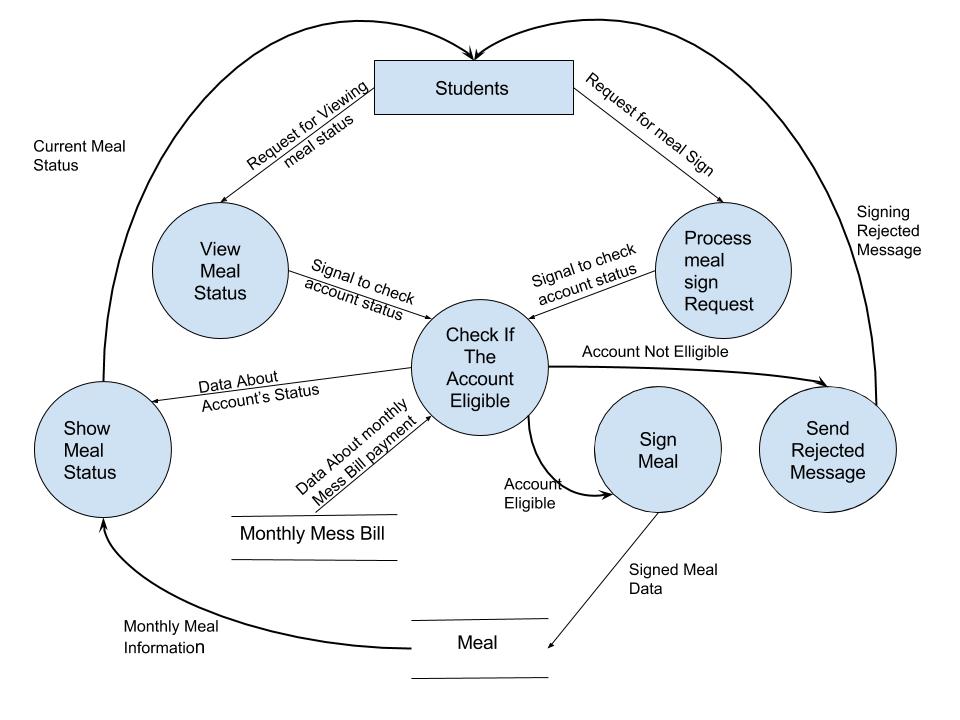
Student can check their bill through our website. The main entity of this dfd is student. At first he collects info for logging in which is his username and password. Then the validity of the collected data is checked. Collect data and validity check are processes here. If the information is valid and student requests for bill, both the mess bill and hall bill can be viewed of anytime through view bill process.

Student can also check their bill if it is correct or not using check bill process. If there is some mistake in the bill, student can inform the authority which is another entity. The adjust extra money process informs authority about extra money. Student is informed by the authority about any changes in the bill status. The view notice process help students to view recent notices from the recent notice table.



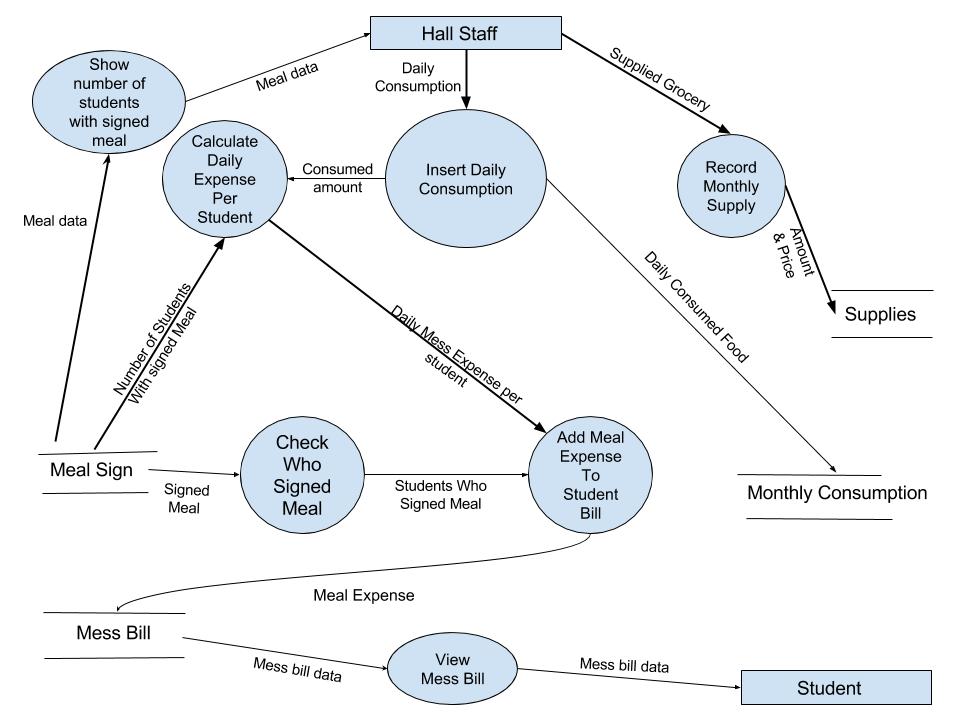
1. Meal sign for students:

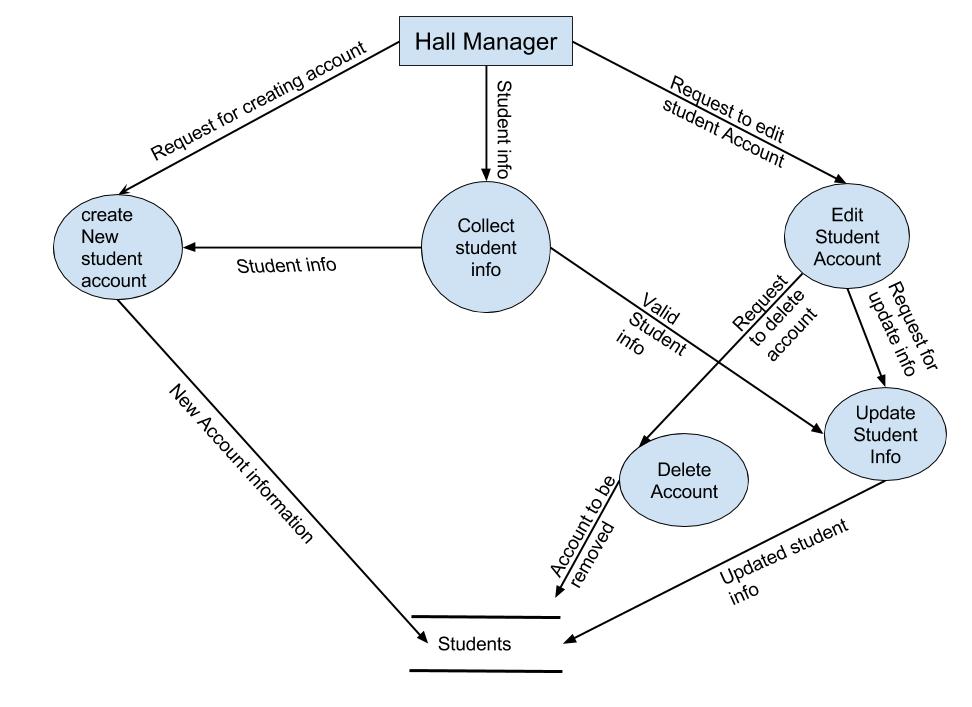
At first the collect info check validity info works as same as bill sign. View meal status let students to view their meal status. If they are eligible to sign meal, the next process helps them sign meal and entry info into the database. The modify info process modifies student information regarding meal.



1. Grocery information input for hall stuff:

The view edit info process allows hall staff to enter student information. Valid data then passes into the student database. Daily consumption data goes to the insert process. Two outputs come out of there. Consumed food and daily consumed food. Consumed food gets stored in the monthly database. Daily consumed food goes through calculate, add meal expense and check process into meal sign database. Monthly bill is stored as payment in mess bill database through insert process. Record monthly supply process records supply. The monthly bill and due and remaining amount is returned to student.





1. The hall manager at first logs in. through checks bill status process he checks bill and then the system goes through change meal status process. Which is saved in student database. The create new account process and update student info process is related to the database. Hall manager uses publish notice process to publish notices.

Class diagram:

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

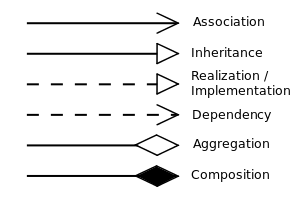
The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a *structural diagram*.

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

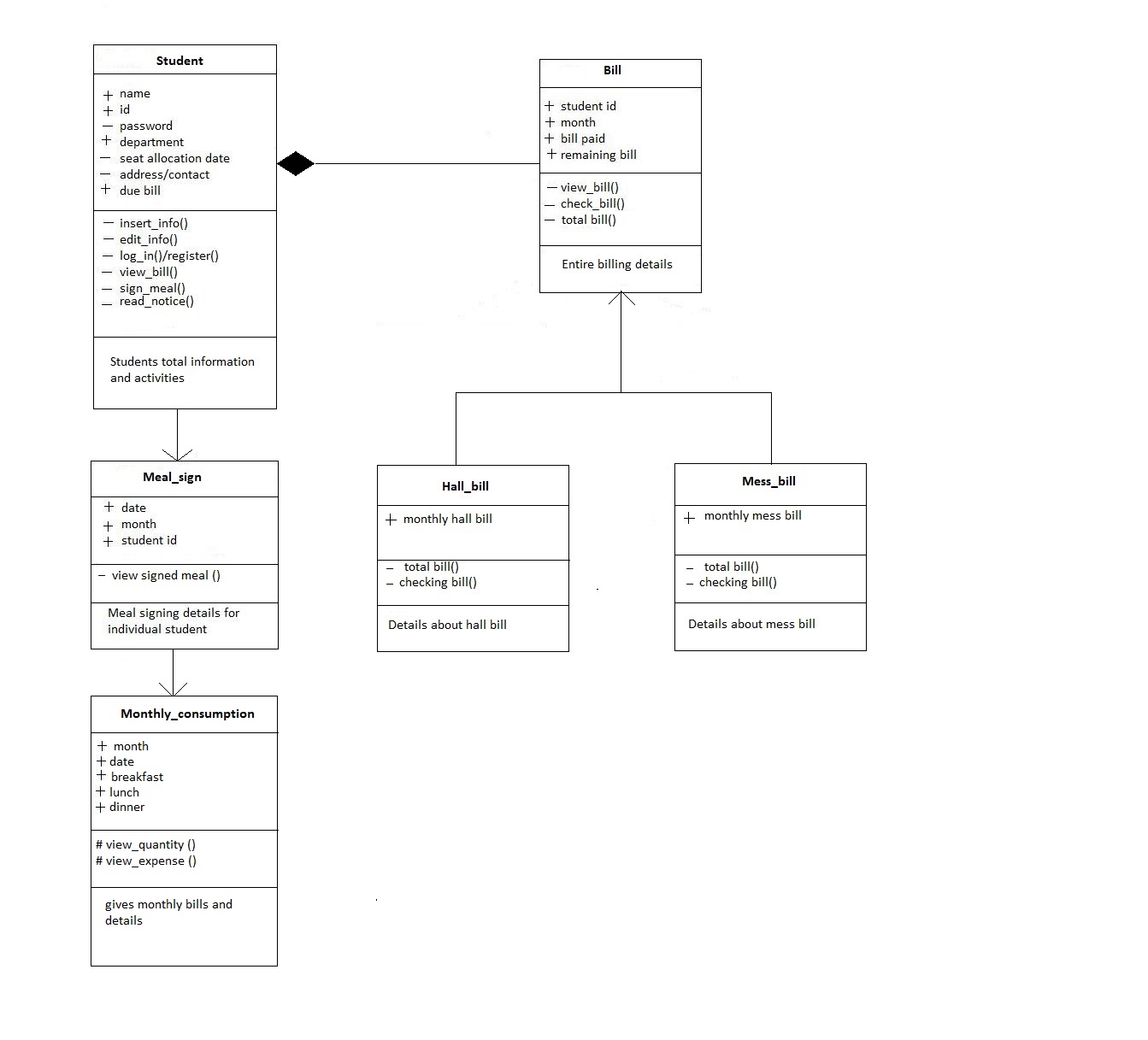
The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.

A relationship is a general term covering the specific types of logical connections found on class and object diagrams. UML defines the following relationships.

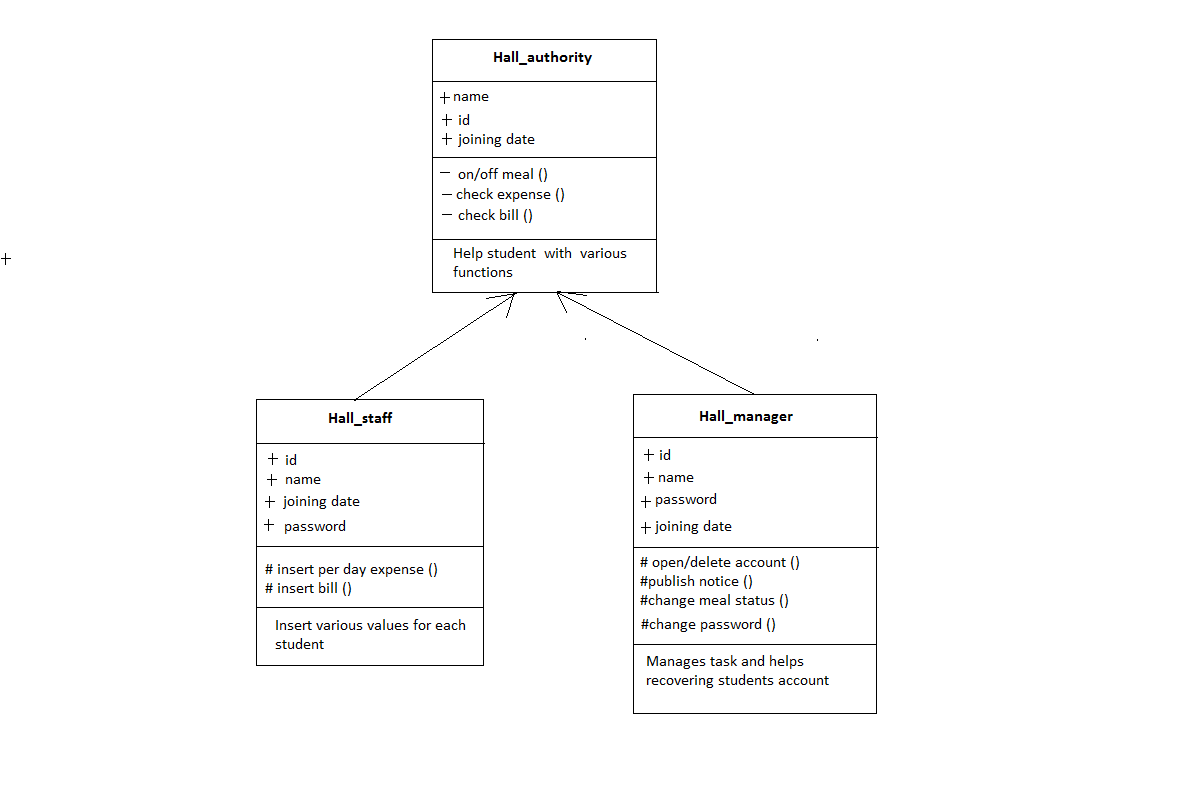


1. The first class diagram shows relation between student and meal sign. The attributes of the student table r added to the classes and the methods are the works which student can do insert, edit, log in, view bill, sign meal, read notice.

the meal sign table inherits student table. It is one to n relation as a student sign lot of meals. The bill table is generalized into two tables hall bill and mess bill which perform their corresponding functions. It is one to one relationship.



1. The second class diagram shows inheritance. Hall manager and staff inherits the attributes of hall authority and perform functions which are separated in class. The supplier table is inherited by monthly consumption as it is dependent on it. It does two functions view quantity and expenses.

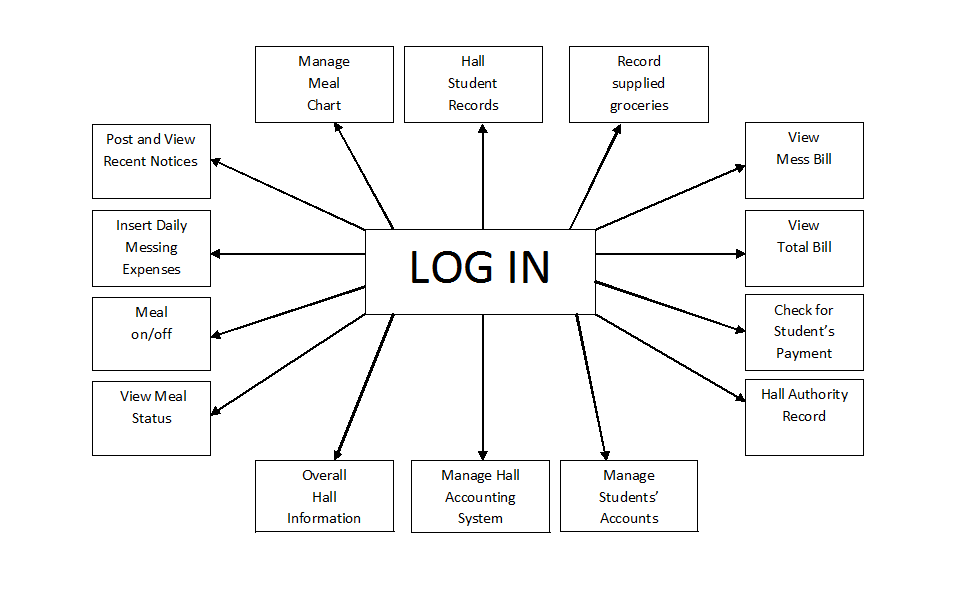


Context diagram:

A system context diagram (SCD) in [software engineering](https://en.wikipedia.org/wiki/Software_engineering) and [systems engineering](https://en.wikipedia.org/wiki/Systems_engineering) is a [diagram](https://en.wikipedia.org/wiki/Diagram) that defines the boundary between the [system](https://en.wikipedia.org/wiki/System), or part of a system, and its environment, showing the entities that interact with it. This diagram is a high level view of a [system](https://en.wikipedia.org/wiki/System). It is similar to a [block diagram](https://en.wikipedia.org/wiki/Block_diagram).

In our project, the account is basically the center of the context diagram is account which represents the system. The entities surrounding it has link to the center. Labelled lines between the entities and the system shows relationship between them.

A Context Diagram (and a DFD for that matter) provides no information about the timing, sequencing, or synchronization of processes such as which processes occur in sequence or in parallel.  Therefore it should not be confused with a flowchart or process flow which can show these things.



**Conclusion:**

We sincerely wish that the software we are about to create will ease out the hall life for students and also suffice the hall management committee to carry out necessary works in a smooth and reliable manner. We hope to expand the project further, on completion of the basic aims with which it will be built and provide our authority with a utilitarian application.

**Reference:**

1. <https://en.wikipedia.org/wiki/class>\_diagram
2. https://en.wikipedia.org/wiki/data\_flow\_diagram
3. Pressman, Roger (2010). *Software Engineering: A Practitioner's Approach*. Boston: McGraw Hill. p. 123. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number)9780073375977.
4. http://buhoz.net/public/books/computacion/desarrollo/requerimientos/Software.Requirements.2e2003.pdf  (page-515)
5. *Software Engineering*, Ian Sommerville, 8th Edition (chapter-7, page-146)
6. *Software Engineering: A Practitioner's Approach* (6th Edition), Roger S. Pressman (chapter-21, page-631)
7. http://energy.gov/sites/prod/files/2014/06/f17/Information\_System\_Owner\_Core\_Competency\_Training\_v01.pdf