HARAMAYA UNIVERSTIY

COLLEGE OF COMPUTING AND INFORMATICS

Software System Design Document

Group Name: ID №:

1. Dibora Lacha 1776/12
2. Betelhem Kibret 1614/12
3. Danawit Yosef 1727/12

Submitted to: Mr. Akubazgi G.

# Introduction

In this document mainly concerned with the design part of application based student cafeteria management system.

System design is the transformation of the analysis model into a system design model. During system design, developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams. The result of system design is a model that includes a clear description of each of these strategies, subsystems and a UML diagrams representing the hardware/software mapping of the system.

# overview of system design

Systems design is therefore the process of defining and developing [systems](https://en.wikipedia.org/wiki/System) to satisfy specified [requirements](https://en.wikipedia.org/wiki/Requirement) of the user.

[Object-oriented analysis and design](https://en.wikipedia.org/wiki/Object-oriented_analysis_and_design) methods are becoming the most widely used methods for computer systems design.

There are two types of system design. These are:-

**Logical design**

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems, designs are included. Logical design includes [entity-relationship diagrams](https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model) (ER diagrams).

### Physical design

The physical design relates to the actual input and output processes of the system. This is explained in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed. In physical design, the following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing requirements,
5. System control and backup or recovery.

The physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system.

# DESIGN DOCUMENT

The purpose of designing is to show the direction how the App is built and to obtain clear and enough information needed to drive the actual implementation of Application. It is based on understanding of the model the application built on system design also focuses on decomposing the system in to manageable parts.

During system design we concentrate on the process of data structures and software and hard ware components necessary to implement it.

## Design goals and objectives

The objectives of designing are to model a system with high quality. Implementing of high quality system depends on the nature of the design created by the designer If one wants to make changes to the system after it has been put in to operation depends on the quality of the system design. So if the system is designed perfectly, it will be easy to make changes to it.

The goal of the system design is to manage complexity by dividing the system in to manageable pieces.

Some of the goals are listed below.

* **Security**: The system should be secured from unauthorized user.
* **Modifiability**: The system should be modifiability to modify different services depending on the need of the institute.
* **Flexibility:** The system able to change to suit new condition or situation.
* **Efficiency:** The system must do what it is supposed to do efficiently without the problem.

# Design goal

Design goals idensstify the qualities that our system should focus on. Many designing goals can be inferred from the requirements of the system. Additionally designing goals are elicited from the users. It is, however, necessary to state them explicitly such that every important design decision can be made consistently following the same set criteria. The system should satisfy the requirements of the user in a technically acceptable level.

The objectives of design are to model the system with high quality. Implementing of high quality system depend on the nature of design created by the designer. If the system needs repair or rebuilding then the whole process will be dependent on system design, so if the whole system is designed effectively and precisely then it is easy to make change in the system.The design goals include:-

Performance criteria:

Performance criteria reflect how the system is expected to behave under normal operating condition. This includes the throughput and memory requirement of the system.

Dependability criteria

These criteria determine how much effort should be expanded in minimizing system crashes and their consequence, and how available the system should.

**Reliability:** The system is expected to accept inputs and requests from the user and make the necessary execution. The systems accept all valid user data and produce the expected output.

**Fault tolerance:** Fault may originate in a system level, in the system environment, or in the interaction between user and the system. Some of the faults that may occur include: computer failure, system failure due to virus, unauthorized user. The system tolerate the error occur due to user-system interaction using proper execution handling methods.

Maintenance criteria

Maintenance criteria highlights how new functionalities can be added to the system in the future.

**Modifiability**: The system should be modifiability to modify different services depending on the need of the institute.

**Portability**: The system works in different plat form and operating system.

**Readability**: The code will be easily readable and understandable by other programmers. To achieve this goal we will supplement the code by proper naming convention, proper indentation and comment.

**Efficiency:** The system must do what it is supposed to do efficiently without the problem.

### Subsystem Decomposition

Subsystem decompositions will help us to reduce the complexity of the system. So the team identifies the following subsystem from the main system:

* User management
* Staff management
* Service Visibility
* Schedule

The user management subsystem control the account of the system user and session management control and manage the period of time which the users spending using the system. The other subsystem is Service Visibility which provide and control Adding meal menu and cancelation of menu and also uses to check the availibilty of meal. The subsystem uses to control and manage addition and deletion of meal menu and the other subsystem is staff subsystem which uses to assign staffs to the cafeteria. The last but not the least subsystem is schedule subsystem which uses to control and manage the schedule of the system.

Manage student info

Manage meal

Manage schedule

Generate report

Manage material

Fig.subsystem de

Fig. Subsystem decomposition diagram

To describe textually as follows:-

1. Manage student info Subsystem

* Order students by their id
* Add new students
* Delete student

1. manage schedule subsystem

* create new food schedule
* update food schedule
* delete food schedule

1. generate report subsystem

* submit report
* receive report

1. Manage meal subsystem

* Prepare meal card
* Update meal card

1. Manage material

* Add new material
* Delete material

**Software Architecture**

Software architecture is all about how a software system is built at its highest level. It provides a basic design of a complete software system. It defines the elements included in the system, the functions each element has, and how each element relates to one another. In short, it is a big picture or overall structure of the whole system, how everything works together.

To form an architecture, the software architect will take several factors into consideration:

* What will the system be used for?
* Who will be using the system?
* What quality matters to them?
* Where will the system run?

The architect plans the structure of the system to meet the needs like these. It is essential to have proper software architecture, mainly for a large software system. Having a clear design of a complete system as a starting point provides a solid basis for developers to follow.

### In our system we have preferred three tier system architecture because of the following main advantages:-

### Scalability – Each tier can scale horizontally

### High Performance – the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.

### High degree of flexibility in deployment platform and configuration

### Better Re-use

### Improve Data Integrity

### Improved Security – Client is not direct access to database.

### Easy to maintain and modification is bit easy, won’t affect other modules .

### In three tier architecture application performance is good.

User interface

System (Infrastructure Platform)

Process(application control

Domain business

Persistent(data)

Data base

Fig-Software architecture

### HARDWARE AND SOFTWARE MAPPING

In this section the team discuss about the overall hardware and software organization of the system so deployment diagram are required to show system hardware/software mapping.

#### Deployment Diagram



Application Server

(Java Application)

Android

JDBC

Figure 22: deployment diagram

The system interface is built by using Java and Database queries. The validation of the system will be done by Java. The system interface which will interact with the database will be programed in JDBC and the databases will be done in MYSQL, and the web server will be run on Apache.

## Persistence modeling /Database design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model, these are the tables and views. In an object database, the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the DBMS.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PK | **Meal**  Meal\_ID  Name  Meal type  Day served  Search | **Data type**  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20) |  | PK | **Student**  Stud\_Id  Fname  Lname  Sex  Password  Status | **Data type**  Varchar(10)  Varchar(10)  Varchar(10)  Varchar(6)  Varchar(10)  Varchar(10) |

|  |  |  |
| --- | --- | --- |
| PK | **Staff**  Staff\_Id  Name  Password  Sex  Email  Address  Phone\_Num | **Data type**  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20) |

|  |  |  |
| --- | --- | --- |
| PK  FK  FK | **Cost Shares**  Payment\_ID  Meal\_ID  Student\_ID  Date | **Data type**  Varchar(10)  Varchar(10)  Varchar(10)  Varchar(10) |

|  |  |  |
| --- | --- | --- |
| PK | **Promotion**  Pro\_ID  Name  Description | **Data type**  Varchar(20)  Varchar(20)  Varchar(20) |

|  |  |  |
| --- | --- | --- |
| PK  FK | **Meal Card**  mealCard\_№  expireDate  Password  studentName | **Data type**  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20) |

**Normalized Database**

**1st normal form**

|  |
| --- |
| **Meal Table** |
| Meal\_ID Name Meal\_type Day\_served Search |

|  |
| --- |
| **Student Table** |
| Stud\_ID Fname Lname Sex Password Status |

|  |
| --- |
| **Staff Table** |
| Staff\_ID Name Password Sex Email Address Phone\_num |

|  |
| --- |
| **Cost Shares Table** |
| Payment\_ID Meal\_ID Student\_ID Date |

|  |
| --- |
| **Meal Card Table** |
| mealCard\_№ expireDate Password studentName |

**2nd Normalization**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Meal Table** | | | | |
| Meal\_ID | Meal\_type | Day\_served | Search |  |

|  |  |
| --- | --- |
| New table | |
| Meal\_ID | Name |

|  |
| --- |
| **Student Table** |
| Stud\_ID Fname Lname Sex Password Status |

|  |  |  |
| --- | --- | --- |
| Cost Shares table | | |
| Payment\_ID | Meal\_Id | Date |

|  |  |
| --- | --- |
| New Cost\_shares table | |
| Payment\_ID | Stud\_Name |

|  |
| --- |
| **Meal Card Table** |
| mealCard\_№ expireDate Password |

|  |  |
| --- | --- |
| New Meal\_card table | |
| mealCard\_№ | Stud\_Name |

**3rd normalization**

**Schedule table**

|  |  |  |
| --- | --- | --- |
| Show table | | |
| Meal\_ID | Show date | Show time |

**Payee student table**

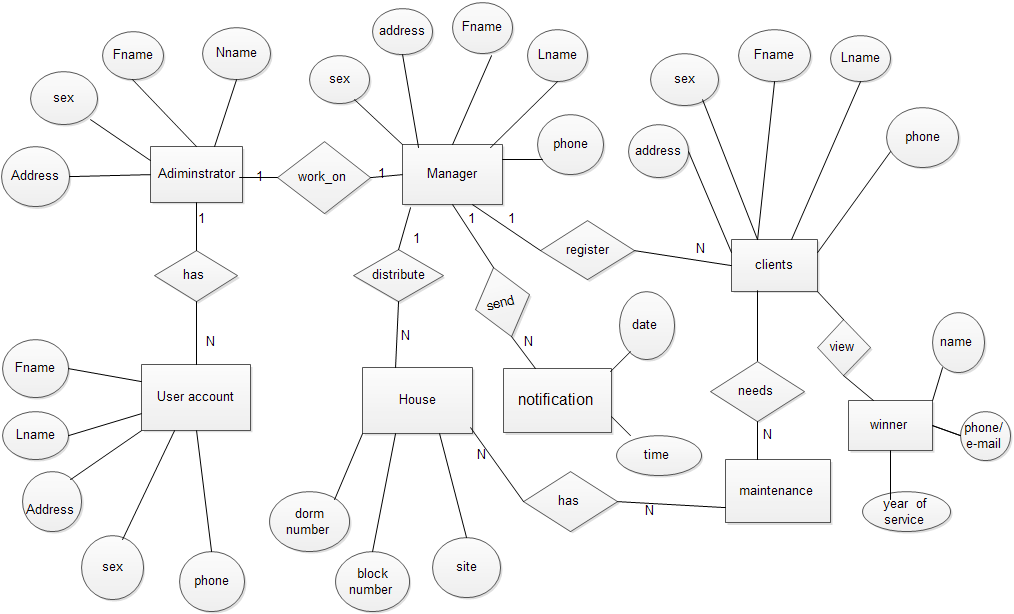
|  |  |  |  |
| --- | --- | --- | --- |
| CostShare payee table | | | |
| Stud\_name | Username | Date | Receipt |

|  |  |
| --- | --- |
| contact table | |
| Email address | Phone number |

**Profile table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Profile table | | | | | |
| Fname | Lname | Gender | Country | city | Password |

|  |  |
| --- | --- |
| Contact table | |
| Email address | Phone number |



menu

meal

Meal\_card

Student

Staff

Fig.1.1 Er\_Diagram

|  |  |  |
| --- | --- | --- |
| PK | **Staff**  Staff\_Id  Name  Password  Sex  Email  Address  Phone\_Num | **Data type**  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20) |

|  |  |  |  |
| --- | --- | --- | --- |
|  | PK | **Student**  Stud\_Id  Fname  Lname  Sex  Password  Status | **Data type**  Varchar(10)  Varchar(10)  Varchar(10)  Varchar(6)  Varchar(10)  Varchar(10) |

|  |  |  |
| --- | --- | --- |
| PK  FK  FK | **Cost Shares**  Payment\_ID  Meal\_ID  Student\_ID  Date | **Data type**  Varchar(10)  Varchar(10)  Varchar(10)  Varchar(10) |

**manage**

**pay has**

|  |  |  |
| --- | --- | --- |
| PK  FK | **Meal Card**  mealCard\_№  expireDate  Password  studentName | **Data type**  Varchar(20)  Varchar(20)  Varchar(20)  Varchar(20) |

**manage**

Fig.1 Relationship Diagram

**Global Control Flow**

Manage Student Information

**Student info list of student**

**List of meal**

Manager

Student

Manage Meal Service

**Meal list**

Schedule

**Schedule delivery meal schedule**

Manage Cost Share

payment Detail Payment detail

## Global Boundary condition

The System Administrator (cafe manager) initiates the web server using the appropriate administrator account that enables him/her to add and modify data available on the system such as, the customer account, student information, notice and others. Of course, the server side is giving 24/7 service unless it gets some problem that makes it down.

Any customer initiates the system to get a connection with the web server using his/her web browser available on the client side. As a result, the home page will be displayed as a boundary object to provide different services and access the pages available there.

In addition to the Home page, the following are some of the boundary objects found in this specific system.

student page, cafe officer page, staff office page.

The system is client–server architecture and allows a remote access.

The following requirements are mandatory on both client and server side.

###### Client Side

* Internet connection should be available on the client side
* Web browser is demanding to connect with the web server of the system
* The customer should be legitimate and having an account provided by the system
* It should give the URL (Uniform Resource Locator) address of the web site
* The customer communicates the different hyperlinks/pages using the homepage

###### Server Side

* The system administrator/Web Master initiates and updates the system using his/her preferred privileges
* The server side should be registered on the local or any other service provider
* It should have Internet connection and database driven-website for remote access

**Access control**

Access control is a security technique that regulates who or what can view or use resources in a computing environment .It is the fundamental concept in security that minimize risk to the organization.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Actor | Staff | food | student | Promotion | Meal card | Cost share |
| manager | Add menu()  View menu()  Delete menu()  Generate report()  View report() | Served() | View() | View()  Add() | Edit() | Calculate() |
| student | View menu() |  | View() | View() | Given() | Pay() |
| employee | View report()  View menu() | Served() | View() | View() | Ticked() |  |

* **Object Design**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **visibility** | **Contract(invariant)** |
| Staff\_id | String | Private | It must Unique  It has 3-6 characters long  Not null |
| Password | String | private | Password not null  Password must be encrypted  It must be 8-15 long character  It should be composed of alphabet,number and special characters |
| Name | String | Public | Name not null |
| Phone | String | private | Phone number not null |
| Address | String | Public |  |

Table 1.1 Attribute specification of staff class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **visibility** | **Return type** | **argument** | **Precondition** | **Post condition** |
| Add menu() | Private | string |  | Menu should not exist | Menu added |
| Delete menu() | Private | string |  | Menu should exist | Menu deleted |
| Generate report() | Private | string |  | Report should exist | Report generated |
| View menu() | Private | string |  | Menu should exist | Menu viewed |

Table1.2 Operation specification of staff class

|  |  |  |  |
| --- | --- | --- | --- |
| **attribute** | **Type** | **Visibility** | **Contract(invariant)** |
| Meal\_id | String | Public | It must be unique  Not null  3-5 character long |
| name | String | Public | Not null |
| type | String | Public | It should be constant |

Table1.3 Attribute specification of food class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **visibility** | **Return type** | **argument** | **Precondition** | **Post condition** |
| Served() | public | string | Meal\_id | Food should exist | Food served |

Table1.4 operation specification for food class

|  |  |  |  |
| --- | --- | --- | --- |
| **attribute** | **Type** | **Visibility** | **Contract(invariant)** |
| Stud\_id | String | Public | Not null  It must be unique for each student  3-6 character long |
| name | String | Public | Not null |
| password | String | Private | It must be encrypted  6-12 character long  It should be composed of alphabet ,numeric and special character |
| sex | Char | Public | Not null  The value must be female or male |
| status | String | Public | Notnull |

Table 1.5 attribute specification for student class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **visibility** | **Return type** | **argument** | **Precondition** | **Post condition** |
| View menu() | public | string | Student name  Stud\_id  password | Menu should exist | Menu viewed |
| Login() | private | string | Username  password | Login page shold exist | Login |

Table 1.6 Operation specification of student class

|  |  |  |  |
| --- | --- | --- | --- |
| **attribute** | **Type** | **Visibility** | **Contract(invariant)** |
| Payment\_id | String | private | It must be unique  Not null  3-6 character long |
| meal\_id | String | public | Not null  unique  3-6 character long |
| Student\_id | String | public | Not null  It must be unique for each student  3-6 character long |
| Date | String | public | Not null  It write it’s own format. |

Table 1.7 attribute specification of cost Share class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **visibility** | **Return type** | **argument** | **Precondition** | **Post condition** |
| Pay() | private | float | Student\_id | Use the service of food | Pay. |

Table 1.8 operation specification of cost share class.

|  |  |  |  |
| --- | --- | --- | --- |
| **attribute** | **Type** | **Visibility** | **Contract(invariant)** |
| Card number | String | private | Not null  Unique  3-5 character long |
| Expire date | String | public | Not null |
| Student name | String | public | Not null |
| Password | String | public | It must be encrypted  Not null  6-12 character long |

Table 1.9 attribute specification of meal card class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **visibility** | **Return type** | **argument** | **Precondition** | **Post condition** |
| Edited() | public | string | card number | The card should exist | Edited |
| Given() | public | string | Sudent id | The card should not exist | Given |

Table 1.10 operation specification of meal card class

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Contract(invariant)** |
| Promotion \_id | String | private | Not null  Unique  3-5 character long |
| Name | String | public | Not null |
| Description | String | public | Not null  Should be valid |

Table 1.11 attribute specification of payment class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | **visibility** | **Return type** | **argument** | **Precondition** | **Post condition** |
| View() | public | String | Promotion\_ id | Promotion should exist | viewed |
| Add() | public | String | Promotion\_ id | Promotion should not exist | Added |

Table 1.12 operation specification of payment class.

**Class Interface**

