# **4.2 Design Goals**

Design goals are targets for design work. These are typically agreed upon by stakeholders as the criteria for comparing design alternatives and evaluating design outcomes. The major goals that we set for this project are listed as follows:

**Availability** - Since what we are building is a webook so it should be available 24 hours a day and 7 days a week (24/7). This means that it should be operable or usable upon demand.

**Maintainability** – while the system is working. Maybe it may face some difficulties which leads the system to process some maintenance procedures. So while designing we target to provide best maintenance system. The system will be easy to add new features including interface and hardware parts to improve its functionality and performance.

**User Friendliness** -A great user interface is often a wonderful design, both user interface and performance should go hand in hand. We designed webook to be more user-friendly it’s very easy, each button or functionality can be seen in the front of the page so a user could not be confused.

**Portability** – The design of the system is easy to mobilize from place to place that enables it to be effective at the time of implementation.

**Cost-effectiveness -**Cost is the number of resources that are going to be sacrificed for the project, whether the cost being in money or materials.

**Reliability** As a design concept reliability is about an application’s ability to operate failure-free. This includes ensuring accurate input and data transformations. To ensure reliability we used the following:-

* -Input data that are more accurate which we found from different reliable researches.
* -Use equipment’s with the best quality
  + Authentication for inserting data so only authorized person could input the correct data

# **4.3 Design trade-offs s**

A trade-off is a situational decision that involves diminishing or losing one quality, quantity, or property of a set or design in return for gains in other aspects. In simple terms, a tradeoff is where one thing increases and another must decrease. [Wikipidea 2016

For a good design that meets all its requirements, the design needs to balance all three factors of technical specification, development cost, and time, taking the following considerations into account from the start:

* Development time
* Available development budget
* Minimizing cost
* User friendly

**Performance vs. Security**

**The tradeoff:** having both high performance and highly secured applications is very costly, that’s why we have to choose one amongst them. Since security-wise, there isn’t that much risk, we have chosen to go ahead with the performance.

The reason we have chosen performance over security is that the system that is being developed is not expected to have very sensitive.

**Portability vs. Development cost**

**The tradeoff:** Do we want the development of a program to be less expensive, or are we willing to pay more to make it work on multiple platforms. Will the cost of developing for multiple platforms will cost more. So, we have decided to minimize cost.

# **4.4 Subsystem Decomposition**

Webook is divided into six subsystems. These systems are divided again into subsystems to make efficient understanding and workflow. We can divide **webook** into admin configuration system, account access, letter management, message and chat system, memo management, and report management system. Let’s look at each system with its functions

1. **Admin configuration subsystem –** This subsystem is controlled by an admin of webook. it serves as the main controlling and configuration center. it is further decomposed into employee management, letter template management, department management, account access management, and company management subsystems. Let’s look at each subsystem to generally know how an admin controlling subsystem works.

* **Company management-** webook operates in a company. Information about the company is recorded when webook runs for the first time. After registration of the company this subsystem edits the company data.
* **Department management-** Companies have departments. In this subsystem, company data is registered, updated, read, and deleted.
* **Employee management-** in this system employee's data are registered, read, updated, and deleted. When an employee is registered a unique id should be provided. This id is given by the company.
* **Letter templater management –** As mentioned in the functional requirement webook provides a template before writing a letter. This template simplifies the process of letter writing. In this subsystem, templates are created, read, updated, and deleted. It also set the approval stage and paths for letters that need approval.
* **Account access management system –** Accounts are created by the user when he/she signup. If passwords are forgotten, it is difficult for the user to access webook. This subsystem reset the password and push the user to enter a new password. It also activates and deactivates user account to prevent unauthorized usage of the system.

1. **Account access –**to use webook, first you have to create an account.Accounts are created by inputting employee IDs and passwords. If an employee ID is unknown it means that the user isn’t registered to the system. He/she should contact the admin. If the employee ID is known the system automatically creates an account. You can login and logout from the system. This subsystem mainly does

* Create an account
* Login and logout

1. **Letter Management –** This subsystem is accessed by an employee or a manager. In this subsystem, you can create, search, edit, update, delete, read and show the progress of letters. You can also set approval stages, select a template and approve letters.
2. **Message and chat-** This system enables us to communicate with texts and files. It has a chat dialog box that is used to write view and send a message or files.
3. **Memo management-** this system creates, search, reads, edits, and deletes a memo.
4. **Report Management-** this system creates, search, reads, edits, and deletes a report.

# **4.5 Design phase models**

## **4.5.1 Class Modeling**

Class diagrams are the blueprints of our system or subsystem. we can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. [Ibm 2021]

System decomposition begins by decomposing the system into cohesive, well-defined subsystems. Subsystems are then decomposed into cohesive, well-defined components. Components are then decomposed into cohesive, well-defined sub-components.[cjs 2012]

In our system(webook) the main components of the system are

* Company
* Department
* Employee
* Account
* Letter
* Message
* Memo and
* Report

Let’s look at each component and draw their class diagrams with their relationship**.**

* **Company** – webook works on the company. So company data should be provided to know which company is webook operates on and also it serves as a header for a letter and a memo. Its attributes and methods are summarized as follows.

|  |
| --- |
| **Company** |
| +name: string,  +logo: string,  +dept\_quantity: number,  +emp\_quantity: number,  + city: string,  +subcity: string,  +woreda: string,  +house: string,  +phone\_1: string,  +phone\_2: string |
| +getCompany():Company  +createCompany():Boolean  +editCompany():Boolean |

* **Department** – in a company there are departments. Based on the organization, Department quantity will differ. Department will be used to indicate which employee belongs to which department. Its attributes and methods are summarized as follows.

|  |
| --- |
| **Department** |
| +id: string,  +name: string,  +phone: string,  + emp\_quantity: number,  +office\_number: number,  +manager:string,  +created\_date: Date |
| +getDepartments():List<Department>  +find(id:string):Department  +search(index:string):List<Department>  +save():Boolean  +edit(id:string):Bolean  +delete(id:string):Boolean |

* **Employee**- in a department there are employees. Its attributes and methods are summarized as follows.

|  |
| --- |
| **Employee** |
| +emp\_id: string,  +first\_name: string,  +middle\_name: string,  +last\_name: string,  +sex: 'M' | 'F',  +position: string,  +salary: number,  +type: string,  +manager: string,  +department: string,  +photo:string,  +phone:string |
| +getEmployees():List<Employee>  +thumbnail(): string  +fullName(): string  +name():string  +find(emp\_id:string):Employee  +search(index:string):List<Employee>  +save():Boolean  +edit(emp\_id:string):Bolean  +delete(emp\_id:string):Boolean |

* **Account** – Employees create an account. its attributes and methods are summarized as follows.

|  |
| --- |
| **Account** |
| +id:string  +password:String  +user\_type::String  +emp\_id:String  +access:'activated'|'deactivated'  +connection\_time:Date  +status:’online’|’offline’  +disconnected\_time:Date  +created\_date:Date |
| +currentUser():Account  +getAccounts():List<Account>  +signUp():Boolean  +Login():Boolean  +search(index:string):List<Account>  +find(emp\_d):Account  +delete(id):Booelan  +update(id):Boolean |

* Letter- an employee can have letters. its attributes and methods are summarized as follows.

|  |
| --- |
| **Letters** |
| id:String,  creater:String,  title:String,  description:String,  draft:boolean  created\_date:Date |
| +inbox(emp\_idstring):list<Letter>  +unread(emp\_id:string):list<Letter>  +outbox:(emp\_id:string):list<Letter>  +search(index:string):list<Letter>  +approval(emp\_id:string):list<Letter>  +find(id:string):list<Letter> |

* Message- an employee can also create a message. its attributes and methods are summarized as follows.

|  |
| --- |
| **Messages** |
| +id:string  +description:string  +sender:string  reciever:string |
| +find(emp\_id:string):Message  +inbox(emp\_id:string):List<Messages>  +unRead(emp\_id:string):List<Message>  +read(emp\_id:string):List<Message> |

* Memo- a memo can also be created by an employee. its attributes and methods are summarized as follows.

|  |
| --- |
| **Memo** |
| +id:string  +description:string  +sender:string |
| +find(id:string):Memo  +inbox(emp\_id:string):List<Memo>  +unread(emp\_id:string):List<Memo>  +read(emp\_id:string):List<Memo>  +outbox:(emp\_id:string):List<Memo>  +searchr(index:string):List<Memo> |

* Report- a report can also be created by an employee. its attributes and methods are summarized as follows.

|  |
| --- |
| **Reports** |
| +id:string  +description:string  +file:string[]  +requester:string  +comment:string  +seen:boolean  +submission\_date:Date  +created\_date:Date |
| +late():boolean  +remaing\_days():number  +find(emp\_id:string):Report  +search(index:string):List<Report> |

These main components can be further classified into sub-components. One way to classify the components is based on the connection an event they have. We drive into two sub-components.

The first component is the **receiver** component this component is driven by an event when a user sends a letter, memo, or report there should be receiver data that contain the user id and other staff that are written below.

|  |
| --- |
| **Receiver** |
| +id:string  +emp\_id:string  +type:”letter”|| “memo”||”report”  +refer\_id:string  +comment:string  +seen:boolean  +seen\_date:boolean  +expire\_date:Date  +created\_date:Date |
| +seen(sid):boolean  +find(emp\_id:string):Reciever  +search(index:string):List<Reciever> |

The second component is the **approval manager** component. This component is driven by when the user sends a letter who need approval. It contains the information of the employee id and other staff that are written below.

|  |
| --- |
| **Approval\_manager** |
| +id:string  +emp\_id:String  +letter\_id:String  +step:Number  +status:”Approved”||”disApproved”  seen:Boolean  seen\_date:Date  approved\_date:Date  comment:String |
| +seen(letter\_id,emp\_ids):boolean  +find(id):Approval\_manager  +isApproved(letter\_id,emp\_id):boolen  +search(index:string):List<Approval\_manager> |

Generally, let look at the class diagram

|  |
| --- |
| **Company** |
| +name: string,  +logo: string,  +dept\_quantity: number,  +emp\_quantity: number,  + city: string,  +subcity: string,  +woreda: string,  +house: string,  +phone\_1: string,  +phone\_2: string |
| +getCompany():Company  +createCompany():Boolean  +editCompany():Boolean |

**References**

**Ibm**(2021).structure class diagram*.ibm.org*. <https://www.ibm.com/docs/en/rsas/7.5.0?topic=structure-class-diagrams>

**Cjs(201)**.System Decomposition. cs.sjsu.edu. <http://www.cs.sjsu.edu/~pearce/modules/lectures/ood/sd/SystemDecomposition.htm>