University of Dhaka

Department of Computer Science and Engineering

Project Report:
Object Oriented Programming Lab(CSE-2112)

Project Name:
Hospice-(The Ultimate Hospital Management System)

Team Members:

Abdullah Ibne Hanif Arean
Roll: FH-12
Registration No: 2019-917-795

Mehadi Hasan
Roll: SH-60
Registration No: 2019-517-843

Ishwor Prasad Dhungana
Roll : IH-63
Registration No:2019-119-430

Hospice-(The Ultimate Hospital Management System)

Introduction

Healthcare is the most critical aspect of our society, and many health care providers face challenges to offer practical and active services to patients. Considering a multispeciality hospital, many people enter and exit the hospital in a day, and maintaining their records safely is tedious. To reduce this type of burden and to manage the financial, hospital administration, and clinical aspects, the Hospital management system came into existence. A Hospital management system will benefit Hospitals or clinics by increasing Processing Speed and Results, Cost Effective, Reduction in Errors, Data Security and Retrieving Ability, Improved Patient Care with Quality and Compliance.

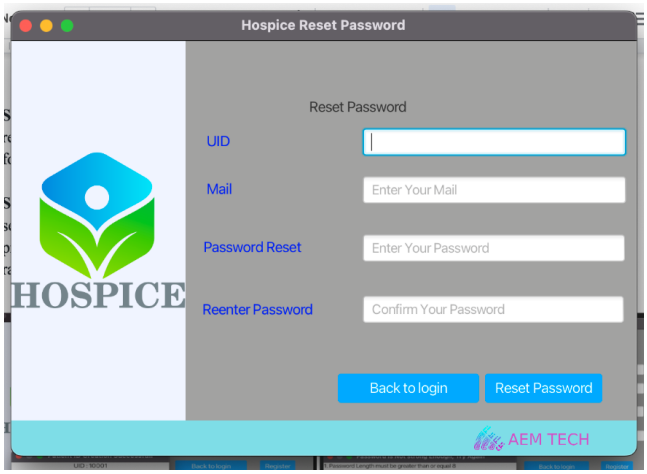
Requirement Analysis & Objectives

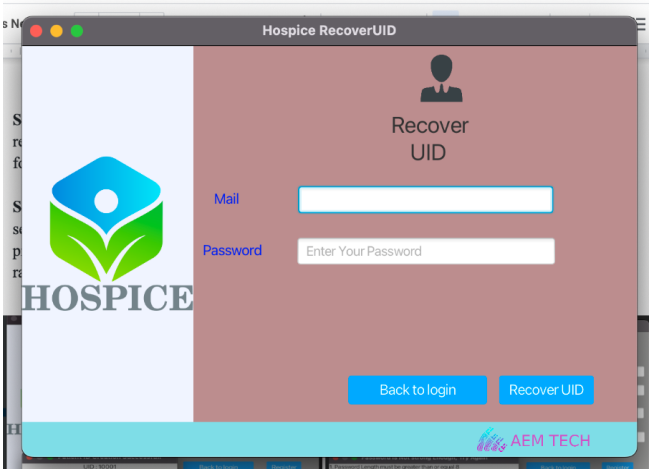
- **Purpose**: The main purpose of this project is the ultimate hospital management system. Actually, there are not many organized management systems in our country's hospitals. Moreover, the systems which exist are neither so interactive to the user nor so secure. So, we have taken the decision to make a project such that we can provide management to the hospitals of our country that is far better secure, easy to use, well-organized, and pretty much interactive to the user. Furthermore, we want to include some features such as a generalized login system, Easy to etc.
- **Intended users:** The project name is suggesting that it will be used in every hospital for the overall management systems. So, everyone connected to that hospital along with all kinds of employees and patients can use the interface of our project. However, in particular, the users will be the Patients, the Doctors, the Admin/Owner, the Nurses, the Lab Assistants, the Pharmacists and the Workers (Cleaner, Gateman etc.).
- **Intensions**: In this project, we want to demonstrate a complete "Hospital Management System". By using this project, a hospital can manage and manipulate its data so much more easily. For instance, managing patients' data, doctor's appointments, manipulating charges, getting prescriptions and other services, multi-lingual facilities, the voice assistance system etc. will be handled in this project. In a nutshell, a complete management system will be found in this project by which sufferings will be lessened for both patients as well as hospital authorities.

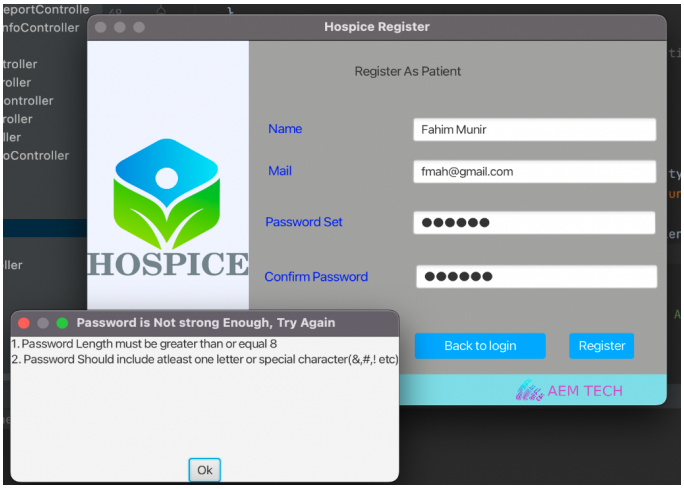
Project Features:

- **Synchronized Multi-User Login System: "Hospice"** offers a platform that integrates logins for seven users (Administrators, Doctors, Nurses, Lab Assistants, Pharmacists, Cleaners, and Patients) onto one page.

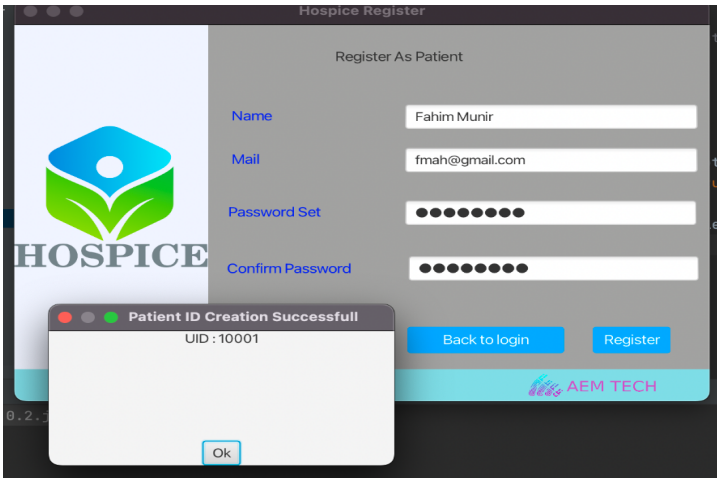
❖ **Secure Password and Seamless Recovery:** As part of our security policy, **"Hospice"** recommends and ensures that passwords combine letters, numbers, and special characters for at least 8 characters, thereby increasing security and reliability.



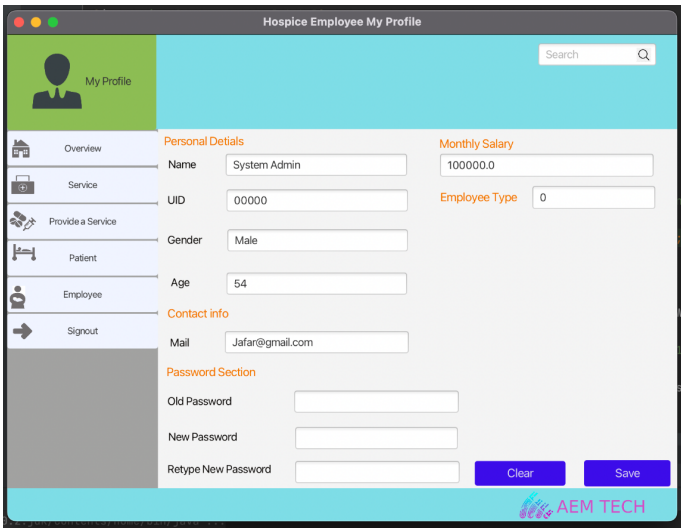




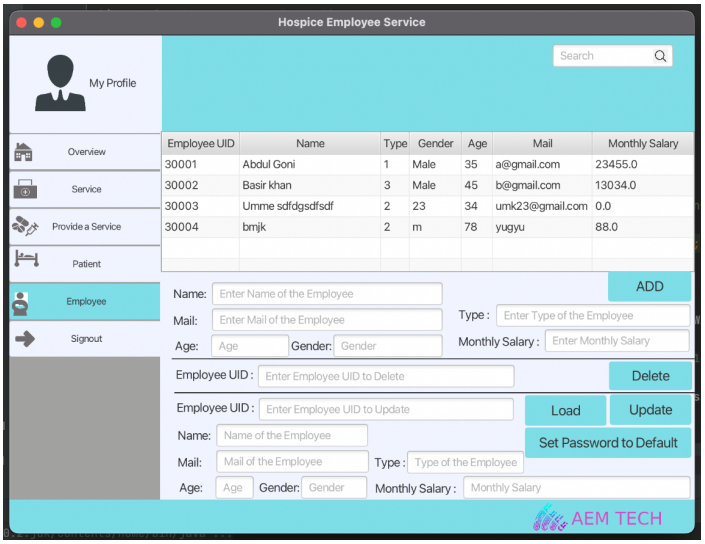
❖ **Simple Registration:** As a new patient, **"Hospice"** makes it easy to register and get the services from the hospital. This is an automatic, fast and straightforward registration process since no employees are needed. For employees, system admin and other high rank employees do the work of registration.

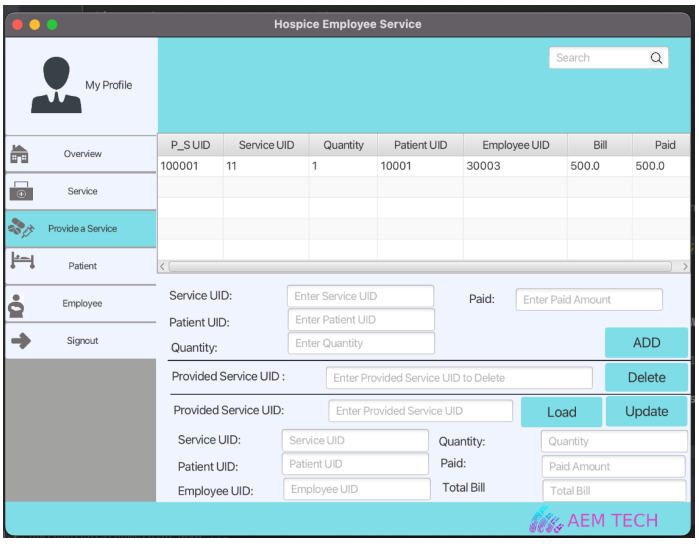


❖ **One Tap to Change Information and Password:** In "Hospice", every user has the ability to change their information and password in the "My Profile" section.



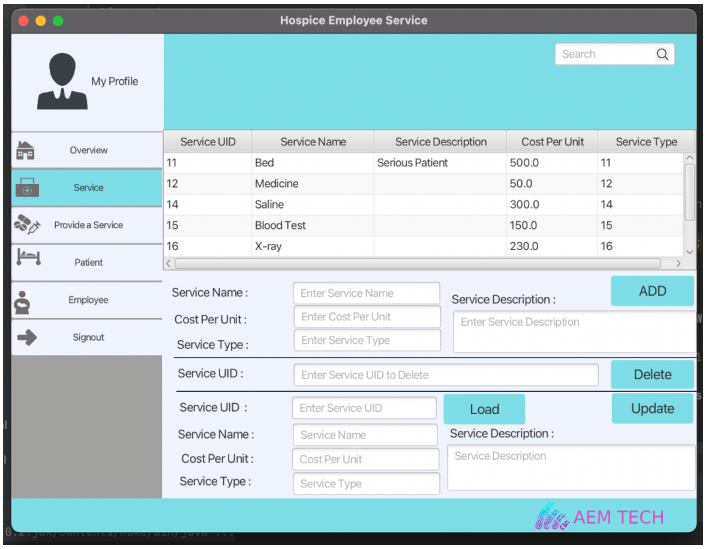
❖ **Fast Accessible Database Integration:** In **“Hospice”** Addition, Deletion, and Update of “users” and “services” are easy and straightforward, with an intuitive and self-explanatory UI(user interface) that

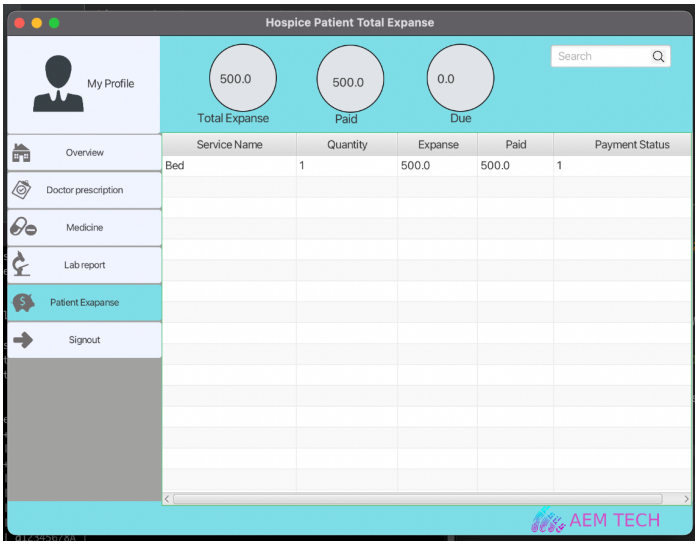




guides the user through the process.

❖ **Complete Billing System** : The patient's billing information is displayed separately in both the employee's and patient's sections. In the patient section there is medicine, lab expanse is





displayed separately along with the total bill. Moreover, payment status is also available in the patient section.

❖ **Unique Identification Number for Each Entity and Service :** Employee as well as services and provided services are identified with different numeric values(i.e. different uid).

❖ **User Input Validness checked :** There pops an alertbox if any undefined data entered by the user.

- **GitHub WorkFlow:** Regularly updated and featurewise work done and project collaboration using github. Extensive commit and cautious merge across multiple platform (Linux, Mac) made the project distinct.

Code Feature:

- Easy but compact Graphical User Interface (GUI)
- Simple and pretty easy to understand the code.
- Uses of Encapsulation, inheritance and polymorphism have been ensured which are the fundamental ideas of OOP language.
- By using encapsulation, we make the data or information much safer.
- By using Inheritance, we have made the best reuse of the code. As a result, we need not write any code more than once.
- Method overriding and method as well as constructor overloading are also used which are the basic concepts of polymorphism. In this case, we used the same name but with different prototype and functionality.
- Used setter and getter methods to set and get various types of data when needed.
- Sometimes, we used constructors to set the data.
- Well-documented code with Software engineering best practices implemented. Ideal for further development.
- Use of Password Encrypted Database for maximum security.
- Different controllers are implemented for different classes along with the fxmls. The controller classes are used for loading the fxmls and controlling the buttons and other intermediate functionalities.
- System admin is hidden and can't be accessed or deleted through the application .
- 50+ interfaces and classes

System Design:

The below page is showing the generalized class diagram of main classes and the interface of the project ‘Hospice’. The first hierarchy shows that four different classes(PatientDBConnectorMySQL, EmployeeDBConnectorMySQL, ServiceDBConnectorMySQL and ProvidedServiceDBConnectorMySQL) have implemented the interface called “ClassDBConnector”. As a result, those four classes had to override the methods(i.e. InsertIntoDatabase(), InsertFromDatabase(), UpdateIntoDatabase()). The next hierarchy is portraying a class “DBLogInManagerMySQL” which implements two different interfaces “LogInManager” and “DatabaseManager”. The interface LogInManager contains four methods: LogInValidate, two overriding methods GenerateUid and ChangePassword. On the other hand, the interface DatabaseManager has only one operation: MakeConnection(). The next hierarchy shows that classes : Service, ProvidedService, Employee and Patient have implemented interface RealEntity. Moreover, these classes also have their own constructors and methods(i.e. setter and getter methods). In addition, class EmployeeType is associated with class

Employee. Here, a point to be noted is that a database is connected to the GUIs in our project. Every interface has given some functionality according to the entity's duty and convenience. Thus a multi-user system has been established.

Class Diagram Link:
https://www.figma.com/file/M8u6wbwkmHTKZdpC7I02SF/Untitled?node-id=3%3A51

Project Modules & Discussion:
Basically, we have five main packages in our project. They are as followings:

1. **Classes Package** : This package includes four interfaces and ten different classes which are considered as the primary classes of the project. These classes will be described later.
2. **Employee Package** : This package has twenty-four controller classes for the "Employee" class within the "Classes Package". Mainly, they are controlling the operations of the "Employee" class of "Classes Package".
3. **LoginRegister Package** : This package contains Login System and its controller classes. Besides, there is a way to recover the uid and reset password.
4. **Patient Package** : This package includes eight different classes by which a patient can see the prescription, bills of different purposes related to the provided services etc.
5. **PopUp Package** : This package contains different types of Alert Box and Confirm Box by which the user is being alerted by warnings on doing something wrong and confirmed by an "ok" message where an operation is fully completed successfully.

Our main classes and interfaces are in the "Classes Package". Let's briefly discuss those classes and interfaces below.

i. **ClassDBConnector(Interface)** :This interface contains three abstract methods which can be inherited by subclasses to perform: data insert into database , data insert from database and data update into database.
ii. **DatabaseManager(Interface)** : It is an interface which has an abstract method for making connection of our code with the database by the classes which are implementing this interface. Basically, this interface is connecting our code with the database.
iii. **DBLogInManagerMySQL** : This class consists of some methods for performing the operations such as LogIn Validation, Generating Uid, Changing Passwords, Admins Privilege to change the password, Checking the passwords' validity etc.

iv. **Employee** : This class contains two Constructors along with some getters and setters to get and set the uid, name, type, gender, age, mail, monthly salary etc. for the employees.

v. **EmployeeDBConnectorMySQL** : This class consists of a method named getPreparedStatement to connect the database with the program to keep the data such as name, type, gender, age, mail etc. for an employee. Besides, there are some methods to be overridden such as InsertIntoDatabase, InsertfromDatabase and UpdateIntoDatabase which are the abstract methods of the interface named "ClassDBConnector". Moreover, there is a method that performs deletion of employee data.

vi. **EmployeeType** : A rough idea of the types of the employees. Basically, their type has been mapped by numbers. There are 6 types of employees.

    1. System admin (mapped by the number 0)
    2. Doctor (mapped by the number 1)
    3. Nurse (mapped by the number 2)
    4. Lab Assistant (mapped by the number 3)
    5. Pharmacist (mapped by the number 4)
    6. Worker (mapped by the number 5)

vii. **LogInManager(Interface)** : This interface contains four (default) methods which can be inherited by subclasses to validate the login system, generate uid and change the passwords.

viii. **Patient** : This class contains two constructors for the patient's information. Moreover,the class has some getter and setter methods to get and set uid, name, type, gender, age, mail, medical history, doctor expanse, lab expanse, Pharmacy expanse, other expenses, total bill etc.

ix. **PatientDBConnectorMySQL** : This class contains a method named getPreparedStatement to connect the database with the program to keep the data such as name, type, gender, age, mail, expenses and total bill etc. for a patient in the database. Besides, there are some methods to be overridden such as InsertIntoDatabase, InsertfromDatabase and UpdateIntoDatabase which are the abstract methods of the interface named "ClassDBConnector". Moreover, there is a method named 'Patientdelete' that performs deletion of a patient's data.

x. **ProvidedService** : It contains two constructors for the provided services related information. Along with that there is a method "generate_bill_type_name" which will return a bill according to the type and name of provided service by considering the quantity of that/those provided service(s). Moreover, there are some getter and setter methods to get and set the uid of provided services, uid of the services, patient uid, employee uid, name of the service, type of the service, quantity, bill, paid status etc. in this class.

xi. **ProvidedServiceDBConnectorMySQL** : This class consists a method named getPreparedStatement to connect the database with the program to keep the data such as uid of provided services, uid of the services, patient uid, employee uid, name of the service, type of the service, quantity, bill, paid status etc. for the provided services in the database. Besides, there are some methods to be overridden such as InsertIntoDatabase, InsertfromDatabase and UpdateIntoDatabase which are the abstract methods of the interface named “ClassDBConnector”. Moreover, there is a method named ‘ProvidedServicedelete’ that performs deletion of the provided service-related data.

xii. **RealEntity(Interface)** : This interface is made for doing different types of intermediate operations.

xiii. **Service** : This class has two different constructors. In addition, there are several getter and setter methods for getting and setting the uid , name, type, cost per unit, discount, description etc. for different services.

xiv. **ServiceDBConnectorMySQL** : It contains a method named getPreparedStatement to connect the database with the program to keep the data such as uid, name, type, cost per unit, discount, description etc. for various types of services in the database. Besides, there are some methods to be overridden such as InsertIntoDatabase, InsertfromDatabase and UpdateIntoDatabase which are the abstract methods of the interface named “ClassDBConnector”. Moreover, there is a method named 'Servicedelete' that performs deletion of service-related data.

Implementation of Design Principles of Java:

In Java, the design principles are the set of advice used as rules in design making. In this project, we tried to implement following design principles:

- **DRY (Don’t Repeat Yourself) :** While developing, we tried to implement a simple yet powerful codebase with the most efficient design, so we used Polymorphism and Inheritance almost in every place if necessary. For instance, in Employee Package Every Class inherits the Class “EmployeeClassController”, Thus we wrote common codes, only once. It made the code easy yet extremely simple for further development.
- **SRP (Single Responsibility Principle):** In our project we used demonstrated naming convention, classes stands for the name it suggests. For instance, “Employee” class is responsible for employee objects and will be changed for only one reason. We tried to use the SRP principle in every class.
- **KISS(Keep It Simple and Stupid Principle):** From naming to implementation, we made our code easy to understand avoiding any kinds of complexity. We divided the code into multiple methods for performing different types of operation so that every operation can be understood easily.

- **ISP (Interface Segregation Principle):** We fully used the interfaces in a class in which the interface is being implemented. For example, the "ClassDBConnector" interface has three functions which are fully used by the classes which implement it.
- **Open/Close Principle:** The code consists of fifty classes with around two hundred functions with self explanatory name and clear goal to achieve. This makes the code open for further development and we don't need to make much change in the existing codebase.
- **Composition Over Inheritance Principle:** Composition refers to an idea where a class can not exist without the other. In our project, the "EmployeeType" class can not exist without the "Employee" class. So, there we have to choose composition over inheritance. Thus our code obeys this principle as well.

Team Member Responsibilities :

Abdullah Ibne Hanif Arean, FH-12(2019-917-795), Team Leader

1. Database Management and Encryption
2. User Interface Design and Class Designer
3. Build System and Version Control

Mehadi Hasan, SH-60(2019-517-843), Team Member

1. Logical hierarchy of classes
2. User Interface and Class Designer
3. Code Debug and Test

Ishwor Prasad Dhungana, IH-63 (2019-119-430), Team Member

1. Graphical User Interface and JavaFX FXML designer
2. Class Designer and Test
3. Documentation of Code

Platform, Library & T +ools

- Java - A high-level, object-oriented programming language used to develop this project
- JavaFX- A software platform for creating and delivering desktop applications!
- Scene Builder - Gluon - provides a visual layout environment that lets us quickly design user interfaces (UI) for JavaFX applications without needing to write any code.
- Intellij - Free and powerful IDE for developing computer software in java!
- Maven- Built-in automation tool used primarily in java projects!
- Inkscape - professional quality vector graphics software and open source!
- Git/GitHub - Software Development and Version Control using Git!

Project Source:

_GitHub Repository: https://github.com/AbdullahArean/Hospice _

_Live Report of the Project: https://docs.google.com/document/d/1DnXGlUN2Cq8pv-_rFWupxkF7x0JYhwmLk4Y8IG9wSps/edit

Limitations:

- Unable to implement multilingual functionality.
- Implementation of voice assistance is yet to be done.
- Could not make the CSS type work that much responsive.
- Database is not portable
- Prescription and Appointment System with unified billing system yet to be developed.
- Pdf/print copy of the bill system yet to be implemented.

Conclusion and Future Work

Apart from learning new languages and technologies, this project taught us collaboration, pressure handling, peer communication, and many other important qualities a software engineer should have. We hope we can implement these learnings in our life in the future to become successful.