UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING
Faculty of Engineering

Semester 2 2023/2024

2a - Problem Analysis and Design

TITLE:
MEDICATION SCHEDULER

LECTURER NAME:
MADAM LIZAWATI

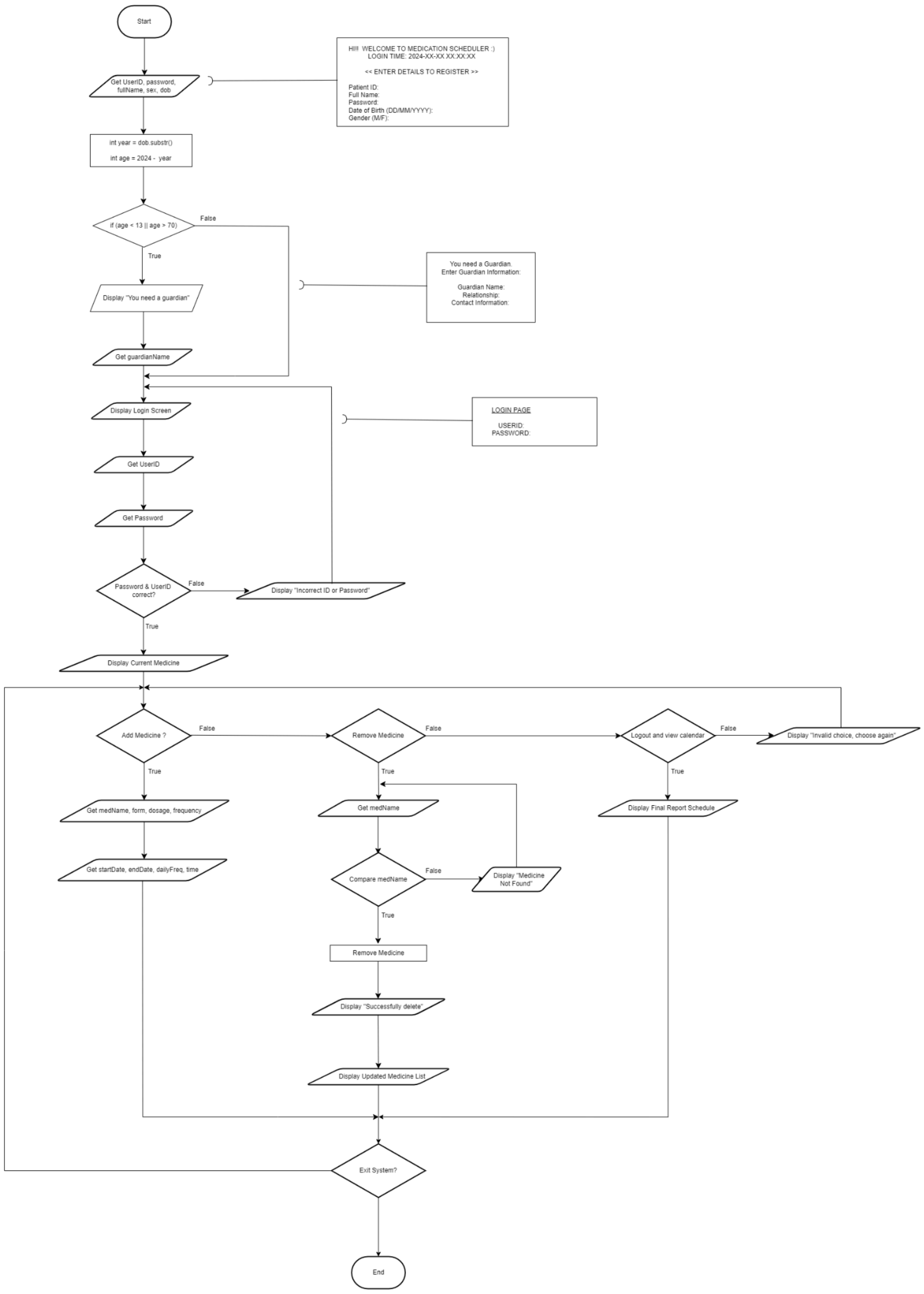
Group Members:

	Name	**Matric Number**
1	ALYA QISTINA BINTI AWALUDDIN	A23CS0041
2	NUR ARISHA BINTI AMYRUL NAIM	A23CS0154
3	TANG JASMINE	A23CS0277

1. FLOWCHART

First of all, a user registration page will be displayed. Users must enter their PatientID, full name, password, date of birth (DOB) and gender (M/F). Then, the system will verify the user's age, if the user's age is below 13 years old or above 70 years old, a new user named "guardian" is needed to monitor the patient's medication status and also serve as a human reminder. Hence, the system will display "You need a guardian" to notify the patient and then get the guardian name. Next, the login screen will be displayed. Users have to enter their registered PatientID and password. The system will authenticate the PatientID and password. If the PatientID and password is incorrect, the system will display "Incorrect ID or Password" to notify the user and return to the login page for the user to re-enter the correct PatientID and password. If the PatientID and password is correct, the user successfully login to the medicine scheduler system.

By default, the system will display the list of current medicine taken by the patient. 3 options will be given. Users can select to (1. Add Medicine, 2. Remove Medicine, 3. Logout and view calendar). By choosing "1", the user has to enter the medication name, form, dosage and intake frequency. The system will also ask users to enter the starting date and end date, daily frequency and intake time of the medication. By choosing "2", the user has to enter the medication name that is intended to remove from the medication list displayed. The system will search for the medication by comparing the medication name input with the medication name in the list. If the comparison fails, this indicates that there's a mistake while inputting the medication name or the medication is not in the list, then, "Medication not found" will be displayed and returned to allow the user to re-enter the correct medication name. If the comparison succeeds, all the information of the specified medication will be removed from the medication list, then "Successfully delete [medication name]" will be displayed. The updated medication list will be displayed. By choosing "3", users get to logout of the system and a report with medication details and schedule will be displayed. If the user has incorrect input while selecting a choice, a message "Invalid choice, choose again will be displayed." and return to allow the user to re-enter the appropriate choice selected. Lastly, users will be allowed to choose whether or not to exit the system, if the user chooses not to exit the system, the system will return to the default display with the medication list and the choices for the user to take action as explained above. If user choose to exit the system, users are able to successfully exit the system.


Start
Get UserID, password, fullName, sex, dob
HI!! WELCOME TO MEDICATION SCHEDULER :)
LOGIN TIME: 2024-XX-XX XX:XX:XX
<< ENTER DETAILS TO REGISTER >>
Patient ID:
Full Name:
Password:
Date of Birth (DD/MM/YYYY):
Gender (M/F):
int year = dob.substr()
int age = 2024 - year
if (age < 13 || age > 70)
False
True
Display "You need a guardian"
You need a Guardian.
Enter Guardian Information:
Guardian Name:
Relationship:
Contact Information:
Get guardianName
Display Login Screen
LOGIN PAGE
USERID:
PASSWORD:
Get UserID
Get Password
Password & UserID correct?
False
Display "Incorrect ID or Password"
True
Display Current Medicine
Add Medicine ?
False
Remove Medicine
False
Logout and view calendar
False
Display "Invalid choice, choose again"
True
Get medName, form, dosage, frequency
Get startDate, endDate, dailyFreq, time
True
Get medName
Compare medName
False
Display "Medicine Not Found"
True
Remove Medicine
Display "Successfully delete"
Display Updated Medicine List
True
Display Final Report Schedule
Exit System?
End

2. CLASSES AND OBJECTS

In our **MEDICATION SCHEDULER** system, we incorporated the concepts we learned in Programming Technique II which is inheritance, association (aggregation and composition) and polymorphism. Inheritance provides a way to create a new class from an existing class. The new class is a specialized version of the existing class. Association on the other hand indicates the relationship between classes. We have two types of association, first is aggregation which means the existence of objects are independent. The other one is composition, which means the existence of the enclosed objects are determined by the enclosing objects. Meanwhile, polymorphism is the ability to perform the same actions differently.

Our system focuses on the patient, dividing it into two categories; regular patient and special patient. Regular patients are adults, with an age range from 13-69 years old. Meanwhile, special patients are kids and old citizens who require a guardian to register. Hence, we have 3 classes for this, with Patient class as parent, and RegularPatient and SpecialPatient as child classes using the concept of inheritance.

INHERITANCE:

CLASSES:
```
Parent (base class):
class Patient {};

Child (derived class) 1:
class RegularPatient : public Patient {};
Child (derived class) 2:
class SpecialPatient : public Patient {};
```

OBJECT:
```
Patient patient;
RegularPatient rPatient;
SpecialPatient sPatient;
```

The **MEDICATION SCHEDULER** system also focuses on the patient's medication, its type and their medication intake schedule. Hence, we implemented a Medication class, with an association relationship of composition with MedType, because MedType cannot exist independently without Medication. The relationship between Medication and MedType is one-to-one, meaning one medicine can only have one type. We also implemented a Report class for patients or guardians to monitor their medication intake. For this, we incorporated the association relationship of aggregation with Medication, because Report can exist independently without medication. However, the report will be blank if no medication is added into it. These two classes are related with a one-to-many relationship, one report can have many medicines in it.

CLASSES:
```
class Medication {}:
class MedType {};
class Report {};
```

OBJECT:
```
int numMed = 0;
Medication *med =  new Medication[numMed];
MedType *medType = new MedType[numMed];
Report report;
```

ASSOCIATION:
```
class Medication {
      private:
      MedType medtype; (composition)


class Report {
      private:
      Medication * med; (aggregation)
```

We also used inheritance concepts alongside polymorphism using class Frequency as the parent, with dailyFreq and weeklyFreq as the child classes. We implement a public method to display the frequency of medicine intake and use it in all of these classes using polymorphism. Then, object Frequency is created with dFreq (daily frequency) and wFreq (weekly frequency) to use it in the main function.

INHERITANCE:

```
Parent class:
Class Frequency {};

Child class 1:
Class dailyFreq : public Frequency {};

Child class 2:
Class weeklyFreq : public Frequency {};
```

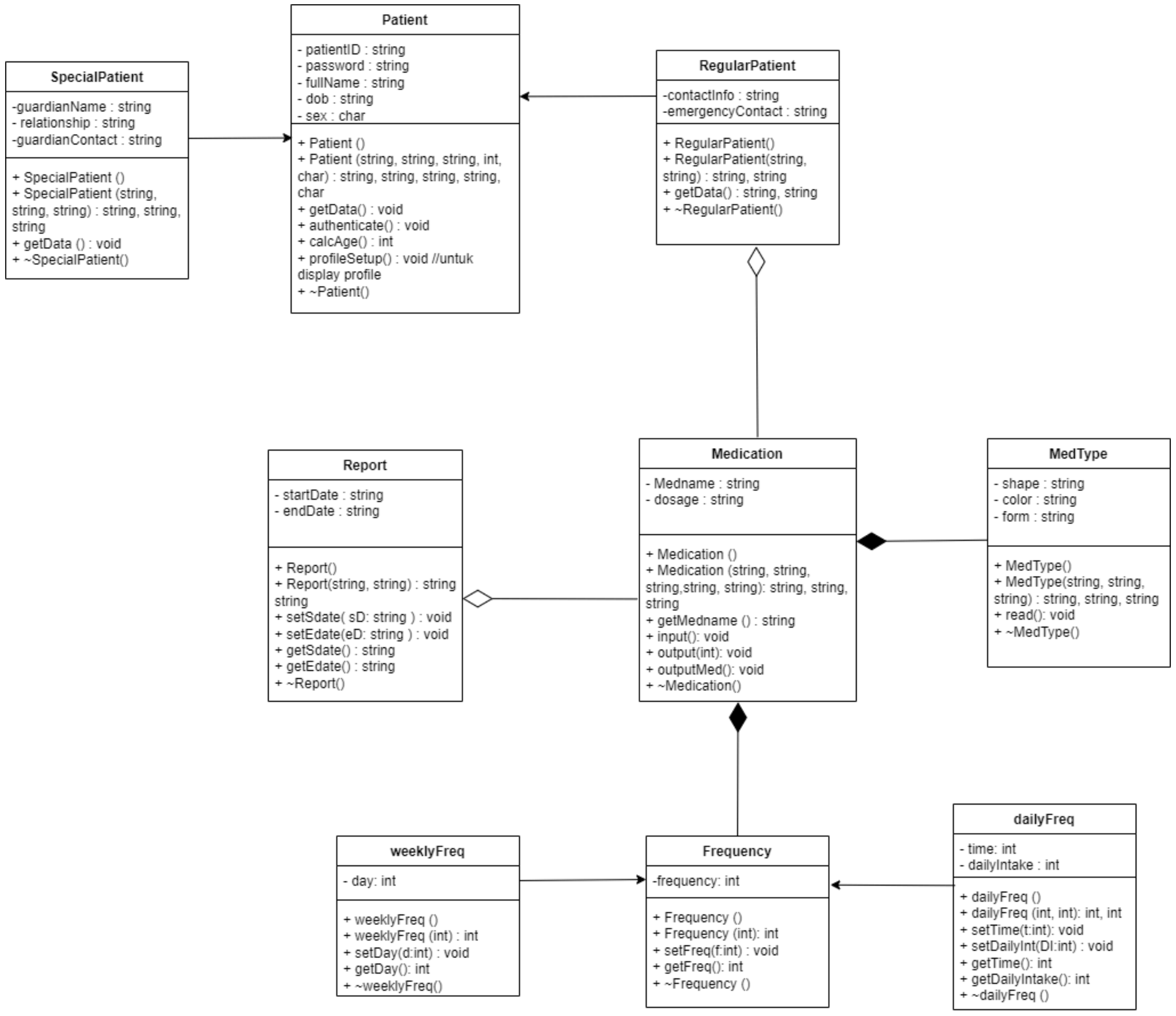
POLYMORPHISM:

```
Class Frequency {
Public:
Virtual void printFreq() {}

Class dailyFreq : public Frequency {
Public:
Void printFreq() override {}

Class weeklyFreq : public Frequency {};
Public:
Void printFreq() override {}
```

OBJECTS:

```
Frequency frequency;
dailyFreq dFreq;
weeklyFreq wFreq;
```

3. UML CLASS DIAGRAM


SpecialPatient
-guardianName : string
- relationship : string
-guardianContact : string
+ SpecialPatient ()
+ SpecialPatient (string, string, string) : string, string, string
+ getData () : void
+ ~SpecialPatient()
Patient
- patientID : string
- password : string
- fullName : string
- dob : string
- sex : char
+ Patient ()
+ Patient (string, string, string, int, char) : string, string, string, string, char
+ getData() : void
+ authenticate() : void
+ calcAge() : int
+ profileSetup() : void //untuk display profile
+ ~Patient()
RegularPatient
-contactInfo : string
-emergencyContact : string
+ RegularPatient()
+ RegularPatient(string, string) : string, string
+ getData() : string, string
+ ~RegularPatient()
Report
- startDate : string
- endDate : string
+ Report()
+ Report(string, string) : string string
+ setSdate(sD: string) : void
+ setEdate(eD: string) : void
+ getSdate() : string
+ getEdate() : string
+ ~Report()
Medication
- Medname : string
- dosage : string
+ Medication ()
+ Medication (string, string, string,string, string): string, string, string
+ getMedname () : string
+ input(): void
+ output(int): void
+ outputMed(): void
+ ~Medication()
MedType
- shape : string
- color : string
- form : string
+ MedType()
+ MedType(string, string, string) : string, string, string
+ read(): void
+ ~MedType()
weeklyFreq
- day: int
+ weeklyFreq ()
+ weeklyFreq (int) : int
+ setDay(d:int) : void
+ getDay(): int
+ ~weeklyFreq()
Frequency
-frequency: int
+ Frequency ()
+ Frequency (int): int
+ setFreq(f:int) : void
+ getFreq(): int
+ ~Frequency ()
dailyFreq
- time: int
- dailyIntake : int
+ dailyFreq ()
+ dailyFreq (int, int): int, int
+ setTime(t:int): void
+ setDailyInt(DI:int) : void
+ getTime(): int
+ getDailyIntake(): int
+ ~dailyFreq ()