# DAYCARE PROJECT

*Object Oriented Design CSYE 6200*
*Group: 05*

# CONCEPTS AND TOOLS

5-Step Ordering Process

**JAVA BACKEND**

**JAVA SWING FOR FRONTEND**

**FACTORY DESIGN PATTERN**

**MVC Architecture**

**DATABSE IN CSV**

# CONTRIBUTIONS

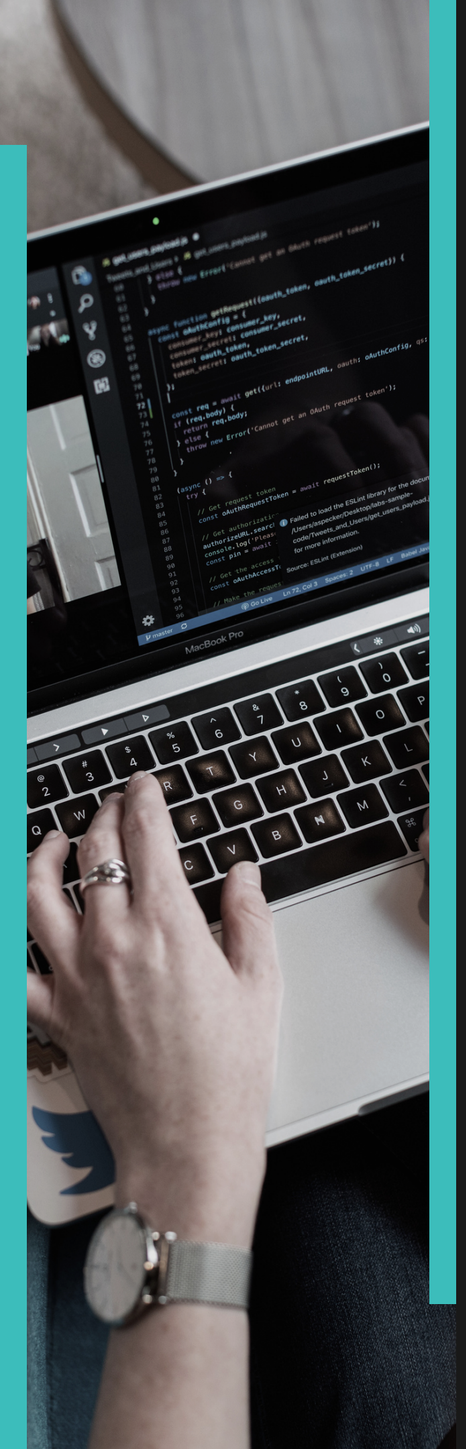
| | | | |
|---|---|---|---|
| Abhinav | Immunization functionality and frontend | Satvik | Classroom functionality and UML |
| Devansh | Teacher functionality and frontend | Tanmay | Student Functionality and UML |
| Kabir | Teacher functionality and documentation | Devanshu | Student and Classroom functionality |
| Gautham | Classroom functionality and documentation | Meet | Immunization functionality and Frontend |
| | | Rakesh | Frontend and Teacher functionality |

# TECHNOLOGY AND DESIGN CONCEPTS

- GENERICS
- RUNNABLE THREADS
- FACTORY PATTERN DESIGN
- LAMBDA FUNCTION
- FUNCTIONAL PROGRAMMING
- INHERITANCE
- POLYMORPHISM
- ABSTRACTION
- ENCAPSULATION
- STREAM API

# LIBRARIES USED

- JAVA SWING
- JAVA UTIL
- JAVA SIMPLE DATE FORMAT
- JAVA AWT
- JAVA UTIL CALENDER
- JAVA TIME
- JAVA.IO
- JAVA NIO
- THREADS
- STREAM



# FUNCTIONALITIES

## TEACHER

- TEACHER REGISTRATION
- TEACHER REVIEW
- VIEW TEACHER DETAILS
- ANNUAL REVIEW ALERT

## IMMUNIZATION

- TRACK IMMUNIZATION
- STORE RECORDS
- UPDATE RECORDS
- UPCOMING DATE

## STUDENT

- STUDENT REGISTRATION
- STUDENT IMMUNIZATION STATUS
- VIEW STUDENT DETAILS
- REGISTRATION RENEWAL
- UPDATE STUDENT IMMUNIZATION

## CLASSROOM

- STUDENT TEACHER MAPING BASED ON STATE RULES

# Generics

```
}

static <T> Collection<List<T>> partitionBasedOnSize(List<T> inputList, int size) {
    final AtomicInteger counter = new AtomicInteger(0);
    return inputList.stream()
                .collect(Collectors.groupingBy(s -> counter.getAndIncrement()/size))
                .values();
}
```

# INHERITANCE

```java
public class Student extends Person {

    private double gpa;
    private String ageGroup;
    private Date registrationDate;


    public double getGpa() {
        return gpa;
    }

    public void setGpa(double gpa) {
        this.gpa = gpa;
    }

    public String getAgeGroup(){
        return this.ageGroup;
    }

    public void setAgeGroup(String ageGroup){
        this.ageGroup = ageGroup;
    }

    public Date getRegistrationDate(){
        return this.registrationDate;
    }
```

# STREAM API

```
public static void viewTeacherInformation() {
    System.out.println("****** Viewing Teacher list ******");
    teacherlist.stream().forEach(System.out::println);
    System.out.println();
}
```

# RUNNABLE THREAD

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
     Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new LandingPage().setVisible(true);
        }
    });
    DayCareController demoObj = new DayCareController();
    demoObj.run();
}
```

# LAMBDA FUNCTION

```
tatic <T> Collection<List<T>> partitionBasedOnSize(List<T> inputList, int size) {
    final AtomicInteger counter = new AtomicInteger(initialValue:0);
    return inputList.stream()
                .collect(Collectors.groupingBy(s -> counter.getAndIncrement()/size))
                .values();
```

# FACTORY DESIGN PATTERN

```
public class StudentFactory {
    private static StudentFactory instance;

    private StudentFactory(){
        instance = null;
    }

    public static synchronized StudentFactory getInstance(){
        if(instance == null){
            instance = new StudentFactory();
        }
        return instance;
    }
```

# THANK YOU