Roskilde Daycare

Our assumptions:

- Open between 6.30 am to 5 pm Monday to Friday. Employees meet in at 6:00 am and prepare for the arrival of children.
- The day care consists of 3 groups of children (maximum 6 children each).
- Each group has 2 teachers at a time (4 teachers for the whole day). The days are split into two shifts - morning shift (6 - 12), afternoon shift (12 - 17).
- Every child must have at least one parent.
- Each employee logs in to the system with their email and password (created when employee is added to the database).

Use cases

Add child to the waiting list

Primary actor: Admin, user

Main success scenario:

- *User is logged in.*
- *User navigates to the waiting list tab.*
- *User selects add button.*
- *User enters basic information about parent and child.*
- *User selects add button.*
- *New child is added to the database.*
- *Confirmation appears.*
- *The system goes back to the updated waiting list.*

Add employee

Primary actor: Admin

Main success scenario:

- *User is logged in with admin credentials.*
- *Admin navigates to the employees tab.*
- *Admin selects add button.*
- *Admin enters basic information about new employee.*
- *Admin selects add button.*
- *New employee is added to the database.*
- *Confirmation appears.*
- *The system updates list of employees.*

Remove employee

Primary actor: Admin

Main success scenario:

- *User is logged in with admin credentials.*
- *Admin navigates to the employees tab.*
- *Admin selects employee.*
- *Admin clicks on remove button.*
- *Employee is deleted from the database.*
- *Confirmation appears.*
- *The system updates list of employees.*

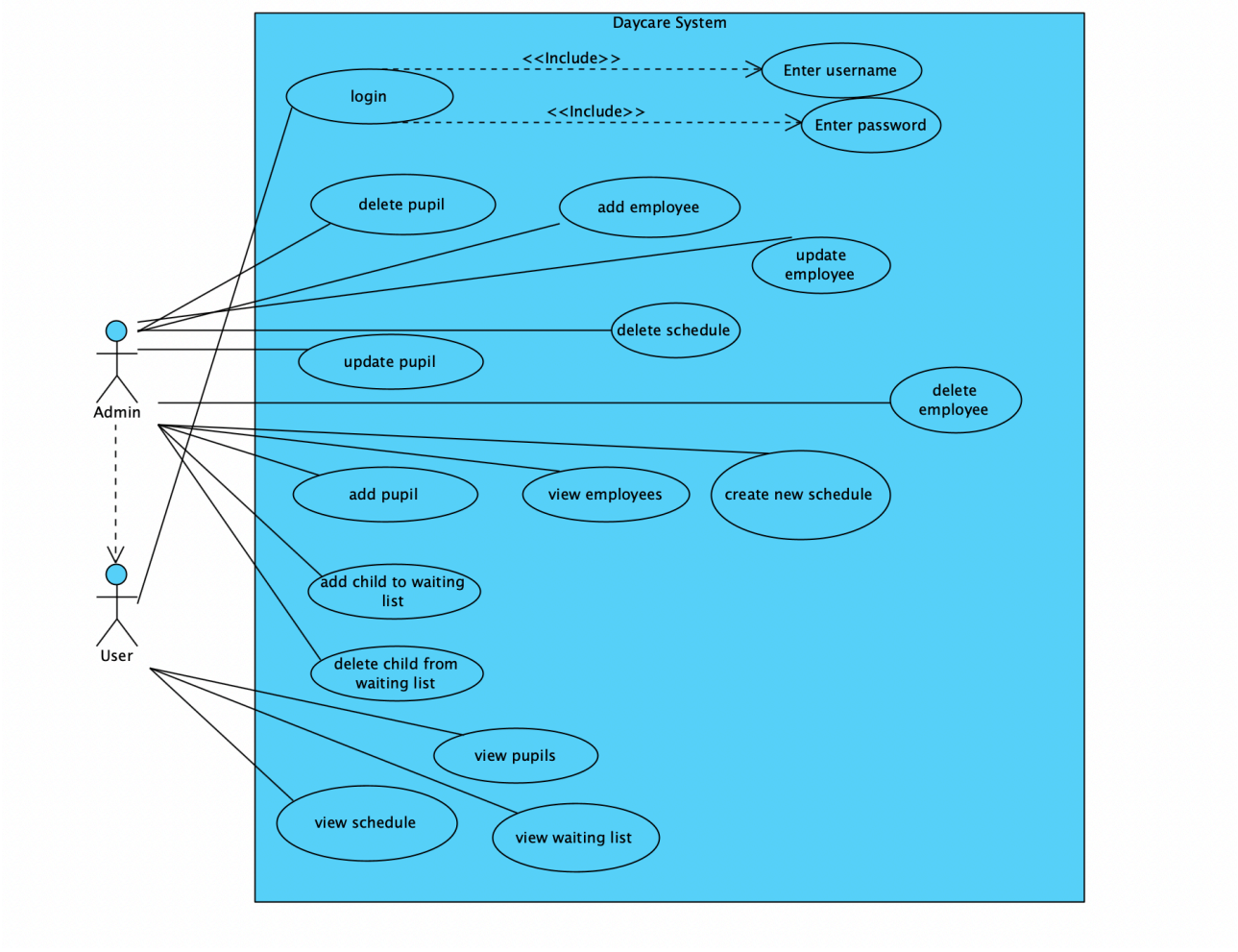
Create schedule

Primary actor: Admin

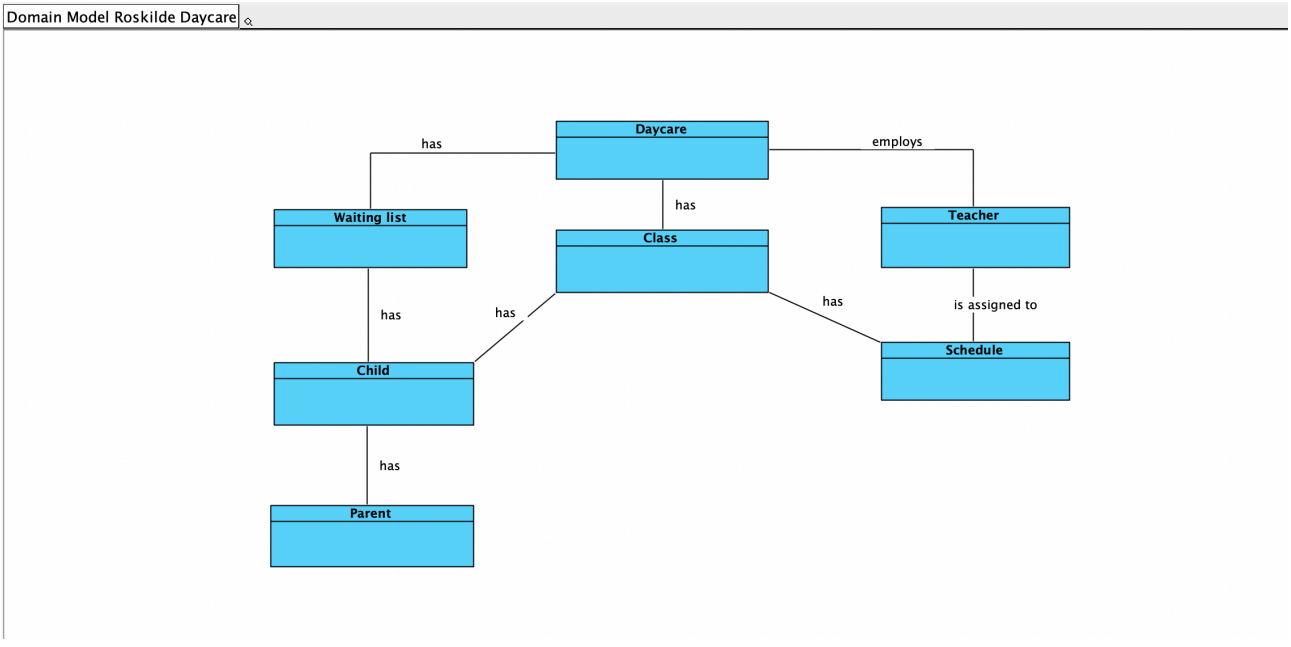
Main success scenario:

- *User is logged in with admin credentials.*
- *Admin navigates to the schedule tab.*
- *Admin selects add button.*
- *Admin assigns certain employees to work on specific days.*
- *Admin saves the information by selecting the add button.*
- *New schedule is added to the database.*
- *The system sets the previous schedule as inactive and new schedule as active.*
- *Schedule tab is updated.*

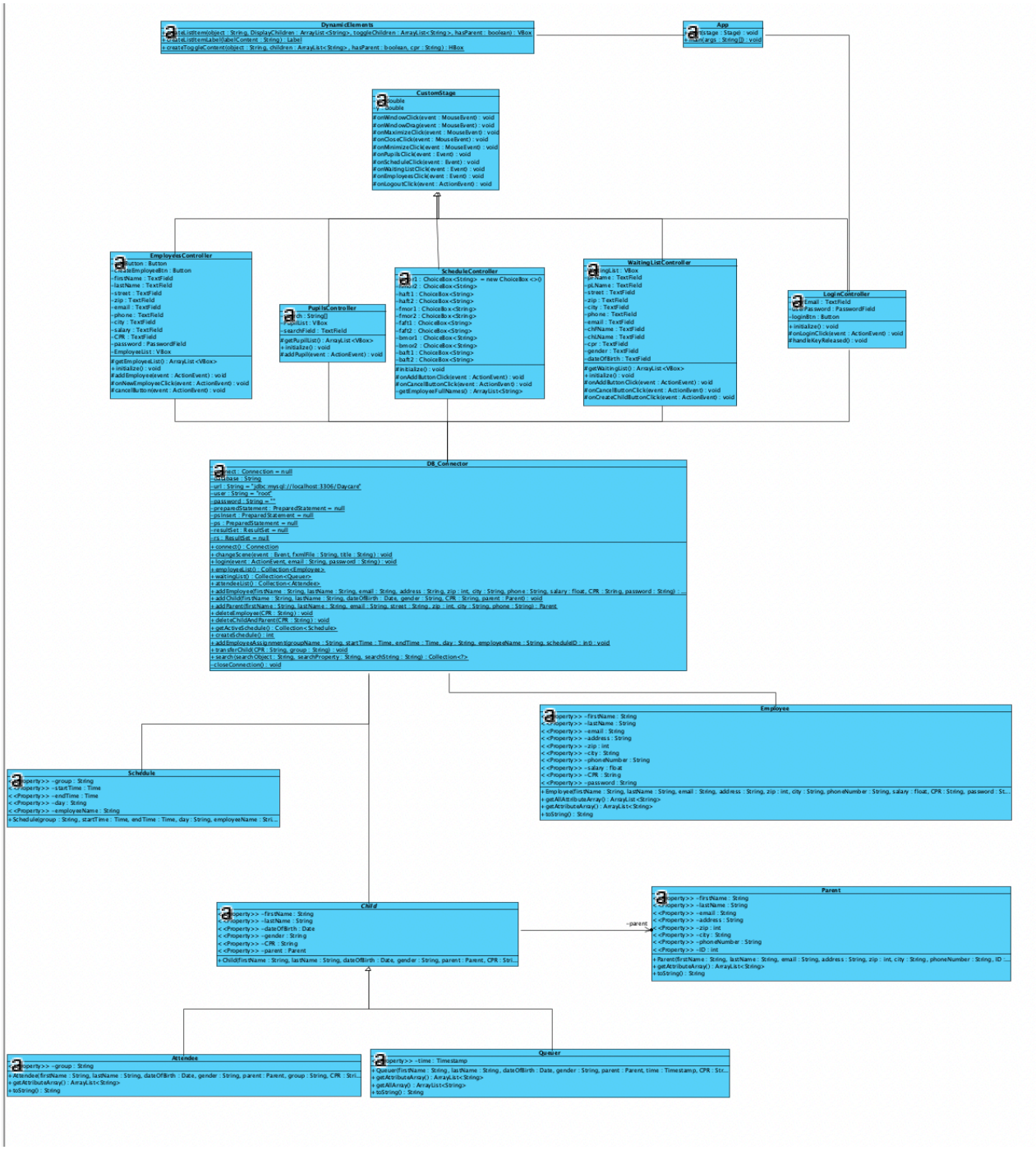
Use case diagram



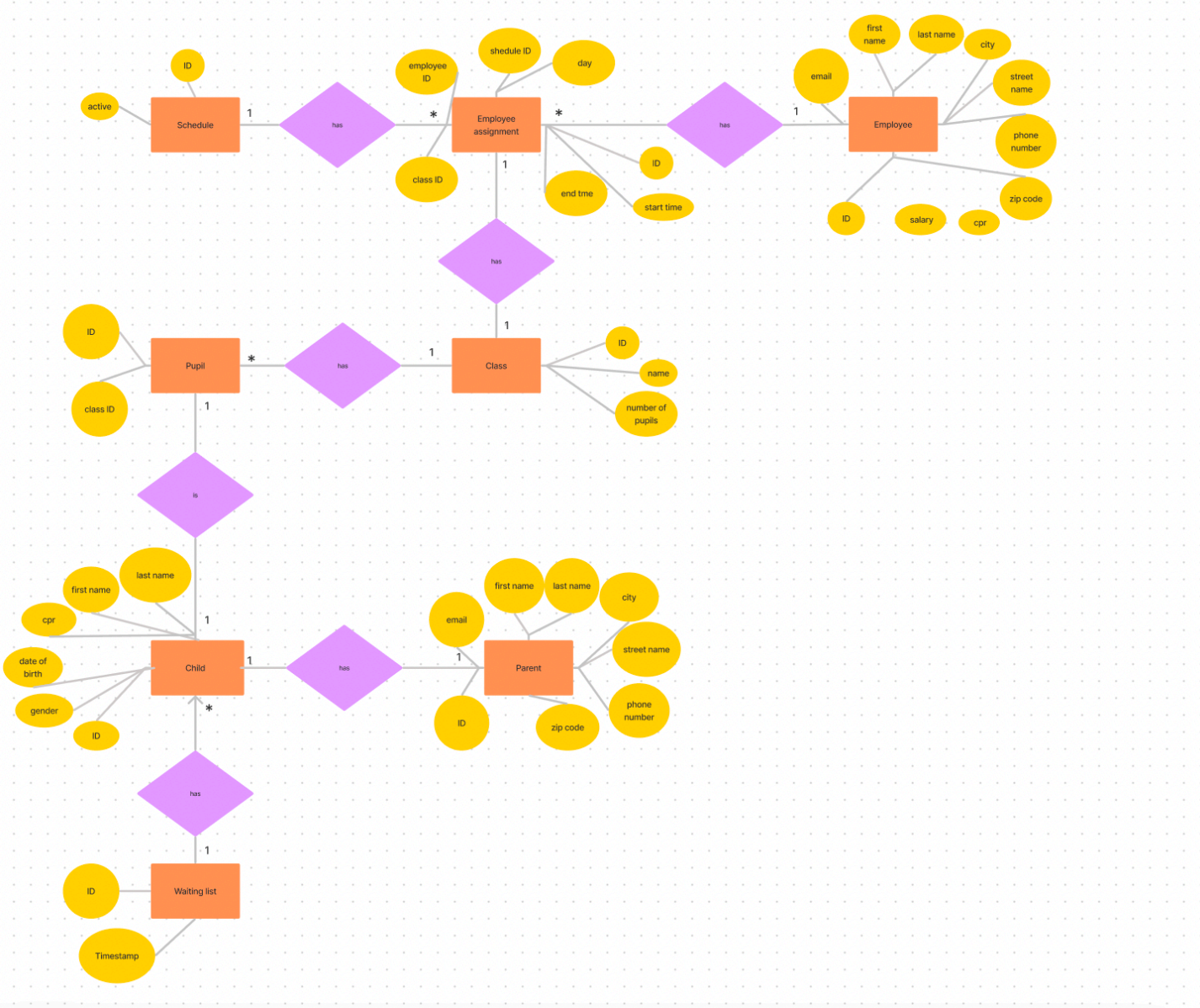
Domain model



Class diagram



ER Diagram


ID
active
Schedule
1
has
employee ID
shedule ID
day
*
Employee assignment
*
class ID
end tme
start time
ID
1
has
1
email
first name
last name
city
street name
Employee
phone number
zip code
ID
salary
cpr
has
1
ID
Pupil
*
has
1
Class
ID
name
number of pupils
class ID
1
Is
1
first name
last name
cpr
date of birth
gender
ID
Child
1
has
email
first name
last name
city
street name
Parent
phone number
ID
zip code
1
*
has
1
ID
Waiting list
Timestamp

Code Snippets

```
    // Login method
    public static void login(ActionEvent event, String email, String password){
        try {
            connect();
            preparedStatement = connect.prepareStatement("SELECT password from Daycare.Employees WHERE
email_address = ?");
            preparedStatement.setString(1, email);
            resultSet = preparedStatement.executeQuery();

            if(!resultSet.isBeforeFirst()) {
                //System.out.println("User not found");
                Alert alert = new Alert(Alert.AlertType.ERROR, "User not found");
                alert.show();
            } else {
                while (resultSet.next()) {
                    String retrievedPassword = resultSet.getString("password");
                    if (retrievedPassword.equals(password)) {
                        changeScene(event, "Pupils.fxml", "Welcome!");
                    } else {
                        // System.out.println("Password didnt match");
                        Alert alert = new Alert(Alert.AlertType.ERROR, "Password doesn't match.");
                        alert.show();
                    }
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection();
        }
    }
```

```
    // Creates employee objects from database and stores them in an Arraylist
    public static Collection<Employee> employeeList() {
        Collection<Employee> employees = new ArrayList<Employee>();

        try {
            connect();
            preparedStatement = connect.prepareStatement("SELECT first_name, last_name, email_address,
street, zip_code, city, phone_number, salary, CPR, password FROM Daycare.Employees");
            resultSet = preparedStatement.executeQuery();

            while(resultSet.next()) {
                String firstName = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");
                String email = resultSet.getString("email_address");
                String street = resultSet.getString("street");
                int zip = resultSet.getInt("zip_code");
                String city = resultSet.getString("city");
                String phoneNumber = resultSet.getString("phone_number");
                Float salary = resultSet.getFloat("salary");
                String CPR = resultSet.getString("CPR");
                String password = resultSet.getString("password");
                employees.add(new Employee(firstName, lastName, email, street, zip, city, phoneNumber,
salary, CPR, password));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection();
        } return employees;
    }
```

```java
 // Method to add employee to the database
    public static void addEmployee(String firstName, String lastName, String email, String address, int
zip, String city, String phone, float salary, String CPR, String password){
        try {
            connect();
            preparedStatement = connect.prepareStatement("SELECT  * from Daycare.Employees where CPR  =
?");
            preparedStatement.setString(1, CPR);
            resultSet = preparedStatement.executeQuery();

            if(resultSet.isBeforeFirst()) {
                Alert alert = new Alert(Alert.AlertType.ERROR, "Employee " + firstName + " " + lastName
+ "already exists.");
                alert.show();
            } else {
                psInsert = connect.prepareStatement("INSERT INTO Daycare.Employees(first_name,
last_name, email_address, street, zip_code, city, phone_number, salary, CPR, password) VALUES (?, ?, ?,
?, ?, ?, ?, ?, ?, ?)");
                psInsert.setString(1, firstName);
                psInsert.setString(2, lastName);
                psInsert.setString(3, email);
                psInsert.setString(4, address);
                psInsert.setInt(5, zip);
                psInsert.setString(6, city);
                psInsert.setString(7, phone);
                psInsert.setFloat(8, salary);
                psInsert.setString(9, CPR);
                psInsert.setString(10, password);
                psInsert.executeUpdate();
                System.out.println("Employee has been added.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection();
        }
    }
```

```java
    // delete child and parent from database, based on child's CPR
    public static void deleteChildAndParent(String CPR) {
        try {
            connect();
            // deletes child with given CPR
            preparedStatement = connect.prepareStatement("DELETE c, p FROM Daycare.Children c JOIN
Daycare.Parents p ON c.parent_ID = p.ID WHERE c.CPR = ?");
            preparedStatement.setString(1, CPR);
            preparedStatement.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection();
        }
    }
```

```java
// Login controller

package com.example.roskilde_daycare;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;

public class LoginController extends CustomStage {

    @FXML
    private TextField userEmail;

    @FXML
    private PasswordField userPassword;

    @FXML
    private Button loginBtn;

    @FXML
    public void initialize(){
        loginBtn.setDisable(true);
    }

    @FXML
    protected void onLoginClick(ActionEvent event) {

        DB_Connector.login(event, userEmail.getText(), userPassword.getText());
    }

    @FXML
    protected void handleKeyReleased() {
        String emailText = userEmail.getText();
        String passwordText = userPassword.getText();
        boolean disableButton = emailText.isEmpty() || emailText.trim().isEmpty() ||
passwordText.isEmpty() || passwordText.trim().isEmpty();
        loginBtn.setDisable(disableButton);
    }

}
```

```java
// method to display children on waiting list

protected ArrayList<VBox> getWaitingList() {
        ArrayList<VBox> list = new ArrayList<VBox>();
        Collection<Queuer> queuers = DB_Connector.waitingList();

        for(Queuer Queuer : queuers) {
            list.add(DynamicElements.createListItem("queuer",Queuer.getAttributeArray(),
Queuer.getParent().getAttributeArray(), true));
        }
        return list;
    }
```

```java
    public static HBox createToggleContent(String object, ArrayList<String> children, boolean
hasParent, String cpr) {
        HBox item = new HBox();
        VBox labelBox = new VBox();
        VBox controlBox = new VBox();

        HBox headingBox = new HBox();
        FlowPane dataBox = new FlowPane();

        labelBox.setPrefWidth(708);

        Label heading = createListItemLabel("Parent");
        heading.setStyle("-fx-font-weight: bold; -fx-font-size: 18px");
        headingBox.getChildren().add(heading);

        if(hasParent) {
            labelBox.getChildren().add(headingBox);
        }


        for(String child : children) {
            Label data = createListItemLabel(child);
            data.setStyle("-fx-font-size: 14px");
            dataBox.getChildren().add(data);
        }
        dataBox.setOrientation(Orientation.VERTICAL);
        dataBox.setVgap(15);
        dataBox.setStyle("-fx-padding: 25px 0 0 0");

        // adding controls to control box
        controlBox.setAlignment(Pos.BOTTOM_RIGHT);
        controlBox.setSpacing(20);

        Button removeBtn = new Button();
        removeBtn.setText("REMOVE");
        removeBtn.setId("remove " + cpr);

        if (object.equals("Employee")){
            removeBtn.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent actionEvent) {
                    DB_Connector.deleteEmployee(cpr);
                    DB_Connector.changeScene(actionEvent, "Employees.fxml", "Pupils");
                    Alert alert = new Alert(Alert.AlertType.INFORMATION, "Employee has been
successfully removed.");
                    alert.show();
                }
            });
        } else if (object.equals("Pupil")) {
            removeBtn.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent actionEvent) {
                    DB_Connector.deleteChildAndParent(cpr);
                    DB_Connector.changeScene(actionEvent, "Pupils.fxml", "Pupils");
                    Alert alert = new Alert(Alert.AlertType.INFORMATION, "Employee has been
successfully removed.");
                    alert.show();
                }
            });
        } else if (object.equals("Queuer")) {
            removeBtn.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent actionEvent) {
                    DB_Connector.deleteChildAndParent(cpr);
                    DB_Connector.changeScene(actionEvent, "WaitingList.fxml", "Waiting List");
                    Alert alert = new Alert(Alert.AlertType.INFORMATION, "Employee has been
successfully removed.");
                    alert.show();
                }
            });
        } else {
            System.out.println("incorrect object passed in to dynamic element");
        }

        System.out.println(removeBtn.getId());
        removeBtn.setStyle("-fx-background-color: #F84E8C; -fx-background-radius: 50px; -fx-text-fill:
white; -fx-font-family: 'Segoe UI Semibold'; -fx-font-size: 14px");
        removeBtn.setPrefWidth(130);
        removeBtn.setPrefHeight(20);

        Button editBtn = new Button();
        editBtn.setText("EDIT");
        editBtn.setId("edit");
        editBtn.setStyle("-fx-background-color: #0075A3; -fx-background-radius: 50px; -fx-text-fill:
white; -fx-font-family: 'Segoe UI Semibold'; -fx-font-size: 14px");
        editBtn.setPrefWidth(130);
        editBtn.setPrefHeight(20);

        controlBox.getChildren().add(editBtn);
        controlBox.getChildren().add(removeBtn);


        labelBox.getChildren().add(dataBox);

        item.getChildren().add(labelBox);
        item.getChildren().add(controlBox);

        item.setStyle("-fx-padding: 25px 50px");

        return item;
    }
```