Roskilde Daycare Project

KEA Copenhagen School of Design and Technology

Lygten 37 - 2200 København N

Computer Science, 2nd Semester 2020

Group 8 : Remi Foss, Aleksandar Miroslavov Minchev, Maja Rebeka Miskéri, Dagmara Przygocka, Cristian-Valentin Purcea

Table of contents

[Project Overview 3](#_Toc36199322)

[Feasibility Study 4](#_Toc36199323)

[**1.** *Operational Feasibility*: 4](#_Toc36199324)

[**2.** *Technical Feasibility*: 4](#_Toc36199325)

[**3.** Schedule: 4](#_Toc36199326)

[***4.*** *Legal:* 4](#_Toc36199327)

[***5.*** *Political Feasibility:* 4](#_Toc36199328)

[***6.*** *Economic Feasibility:* 4](#_Toc36199329)

[ 4](#_Toc36199330)

[Description Of Development 5](#_Toc36199331)

[The Database 6](#_Toc36199332)

[Java Development 7](#_Toc36199333)

[Snippets Of Code 10](#_Toc36199334)

[Appendices 14](#_Toc36199335)

[FURPS+ 14](#_Toc36199336)

[Diagrams 15](#_Toc36199337)

[Class Diagram 15](#_Toc36199338)

[Domain Model 15](#_Toc36199339)

[Use-Case Diagram 16](#_Toc36199340)

[Database 17](#_Toc36199341)

[ERD 17](#_Toc36199342)

[EERD 18](#_Toc36199343)

[SQL Snippets 19](#_Toc36199344)

# Project Overview

**Introduction**

In this project our team created administrative IT-system to for Roskilde Daycare. The main purpose of the system is to aid in maintaining the daycare’s administrative tasks in a simple manner.

Some parts of the code & functionality will be presented and explained here, although, for a better comprehension of the project, we recommend checking out the GitHub Repository :

* https://github.com/CristiPV/Roskilde-Daycare-Project.git

**Requirements**

Here is a list of the functional requirements we discussed at the beginning of the project :



Later on, we conducted a deeper analysis of the projects requirements and we created a FURPS+ Model, which can be found in the Appendices section, or by clicking [here](#_FURPS+).

**System description**:

*Available features*:

* add, remove, display, child
* add, remove, display teacher
* add, display parent
* add, delete, display schedule
* add activity to the schedule
* add, display activities
* add, delete, display group,
* add child to the group,
* add, delete, display waiting list
* add, delete, display invoice
* add, delete, display appointment

*Data storage*:

All the data used by the System is being stored on an online Database hosted by the Amazon Web Service platform.

Description:

In order to create the system we used Java programming language to create functionalities and MySQL to deliver necessary information from database.

In the system there are two types of users:

-Administrator : there is only one Administrator account which has access to all functionality of the system.

-Teacher : there is several accounts of Teacher user and more can be created by the administrator.

Teacher account has access to functionalities like:

* add, delete, display waiting list,
* add, delete, display appointments,
* add, delete, display groups,
* display schedule,
* display phone list

# Feasibility Study

1. *Operational Feasibility*:
   * The system complexity is quite low, allowing the potential users to access key functionality it with ease.
   * The project will aim to solve the business’ problems by reducing the time required to maintain administrative tasks.
2. *Technical Feasibility*:
   * The team will be able to make the system work.
   * The technology required for the completion of the project is available and ready to be used : Amazon Web Services for hosting the database.
   * All members of the team are sufficiently skilled to see the project through to its completion
3. Schedule:
   * The project is estimated to be finished before the given deadline.
   * The team’s focus is to create the minimum viable product with the necessary functionality in order to ensure a product can be delivered before the deadline.
   * There will be daily meetings to assign tasks and check progress on already existing tasks.
4. *Legal:*
   * GDPR - storing of the people’s data such as birthdates, etc. Therefore, we offer an ability for Parents to request information on what data we store about them as well as complete deletion from the system.

All users have to give consent for storing the personal information when they join our Daycare system.

1. *Political Feasibility:*
   * The key stakeholders : Marianne, Douglas & Cay approve of the project and support it.
2. *Economic Feasibility:*



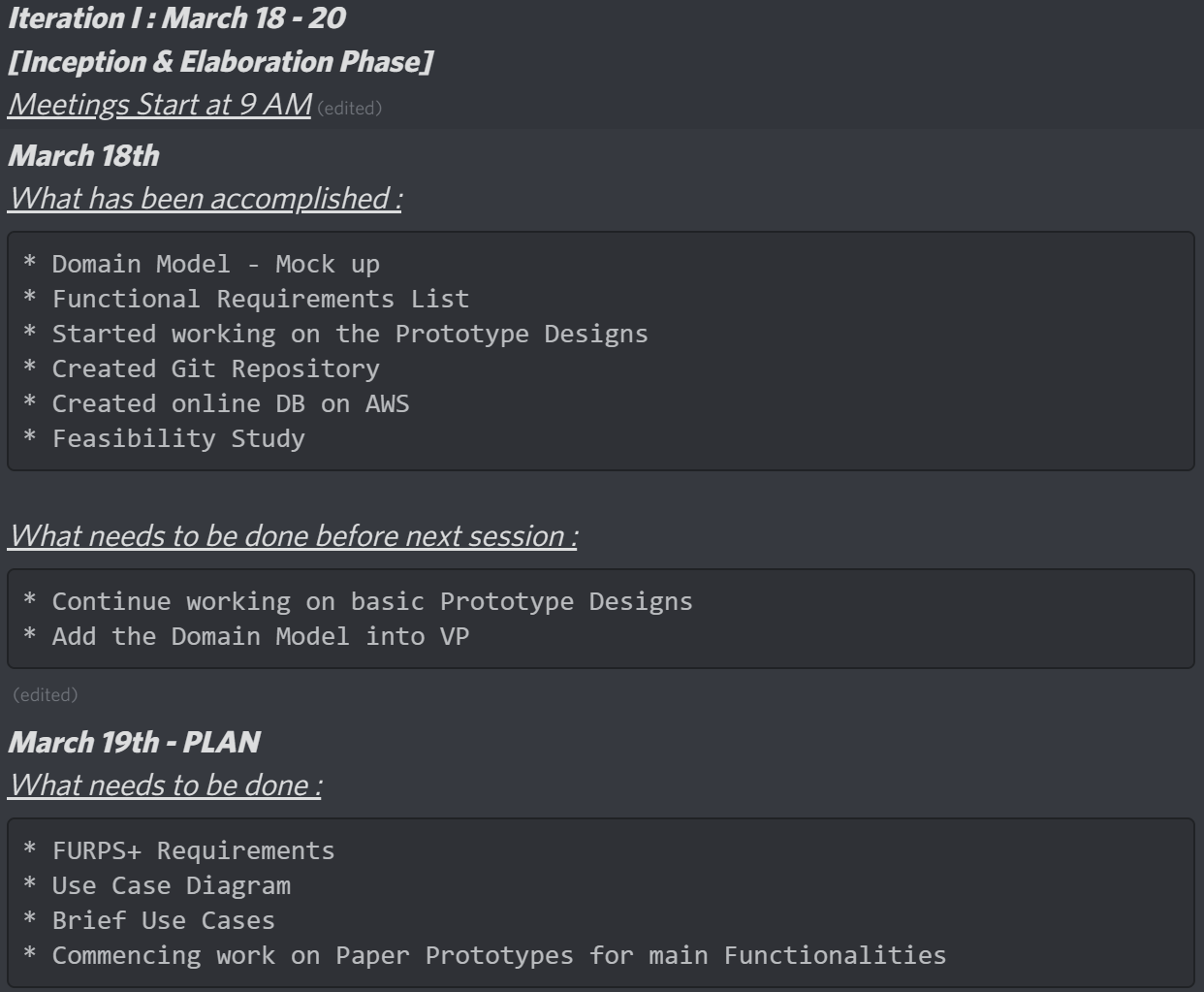
# Description Of Development

The development of this project was done in iterations, following the UP model ( to the best of our abilities ), with the help of the following tools :

Discord – Managing the team and the progress + Arranging meetings.



For keeping track of what needs to be done each day + what has been covered each day, as well as the current progress within each iteration, we used a channel in Discord, where an agenda was maintained (more or less daily) :



The Development started on March 18, with the first Iteration, planned to cover the Inception Phase. However, due to the small scale of the project, we ended up covering the Elaboration Phase as well.

We determined the requirements, done the feasibility study to determine whether the project was worth completing ( we would have completed it anyways ) and started creating diagrams, such as the Domain Model & Use Case Diagram.

Paper Prototypes have been done in order to establish how the flow of the UI should be like, as well as what other requirements and functionalities need to be added.

At the end of this Iteration, we also created the first version of the Class Diagram ( which had to be completely redone once we discovered how big the impact of interacting with a database was on the structure of the code ).

The following Iteration ( Construction Phase ) started on 21st of March and was devoted solely to coding the MVP ( Minimum Viable Product ).

One part of the team focused on the Database creation & population ( with realistic data & information );  
In the meantime, the other part of the team focused on creating the structure of the Java code in IntelliJ and preparing for writing the actual functioning queries on the DB.

The next step was to update the Class Diagram to better reflect the new Structure of the program we had in mind.

After all of us understood how the connection to the Database is being done, as well as how to write actual queries that affect & retrieve information live from the DB, we started defining what the queries should look like so that we would achieve the desired result & we coded them, bringing this Iteration to an end.

The final Iteration, which comes with the Deployment Phase, started on the 24th of March and mainly consisted of bug-fixing, updating the CLI & overall making sure that the program works properly.

All Diagrams can be seen in the [Appendices Section](#_Diagrams).

# The Database

In order to create the Database, we first designed a quick ERD ( Entity Relationship Diagram ) so that we can understand how the DB should look like.

Later, based on the ERD we later created an EERD ( Enhanced Entity Relationship Diagram ) in MySQL Workbench to properly showcase all entities and the relationships between them.

The DB creation code was then generated by MySQL Workbench ( based on the EERD ) and then had to be edited to better suit our needs.

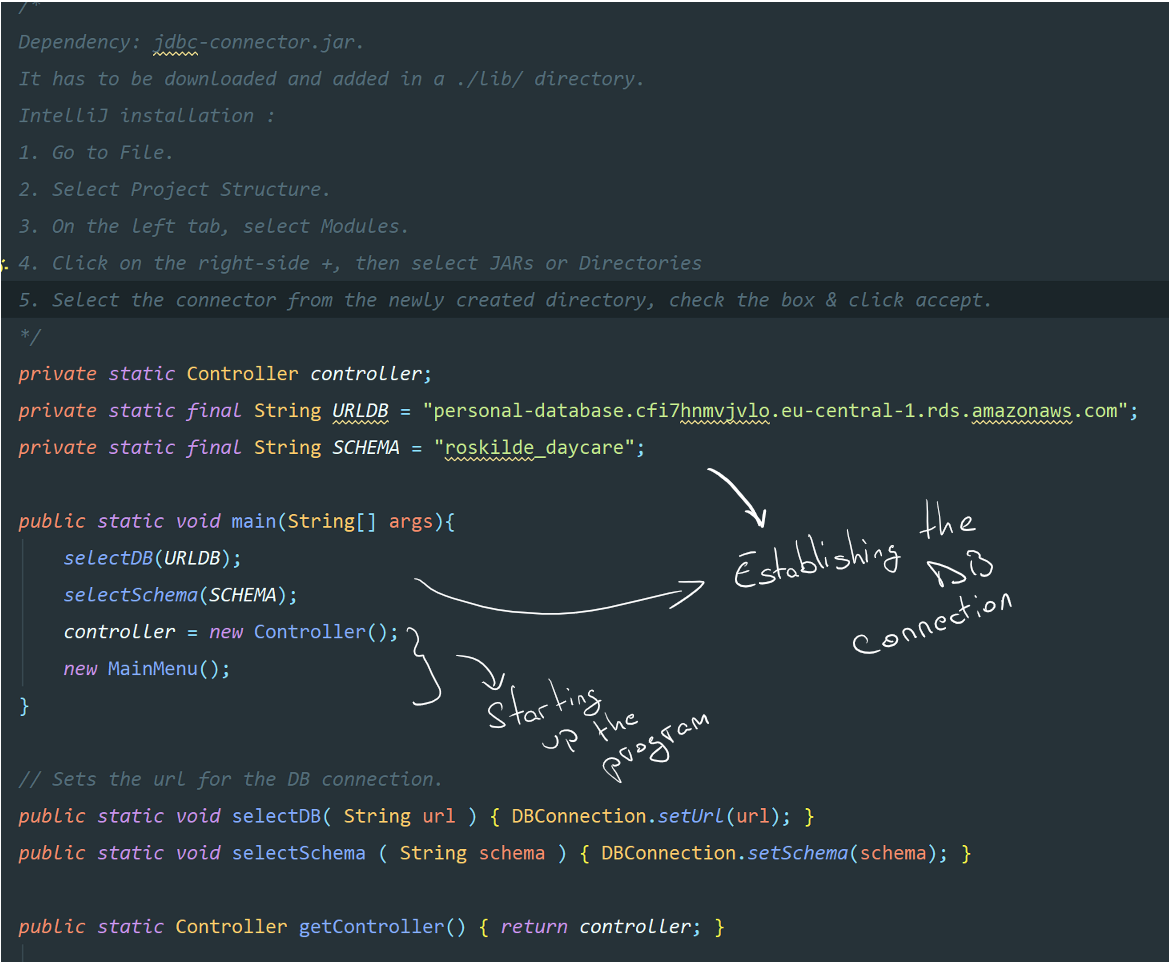
After creating the DB, we worked on populating it with realistic data, which helped in developing a proper vision on how the data should be used.

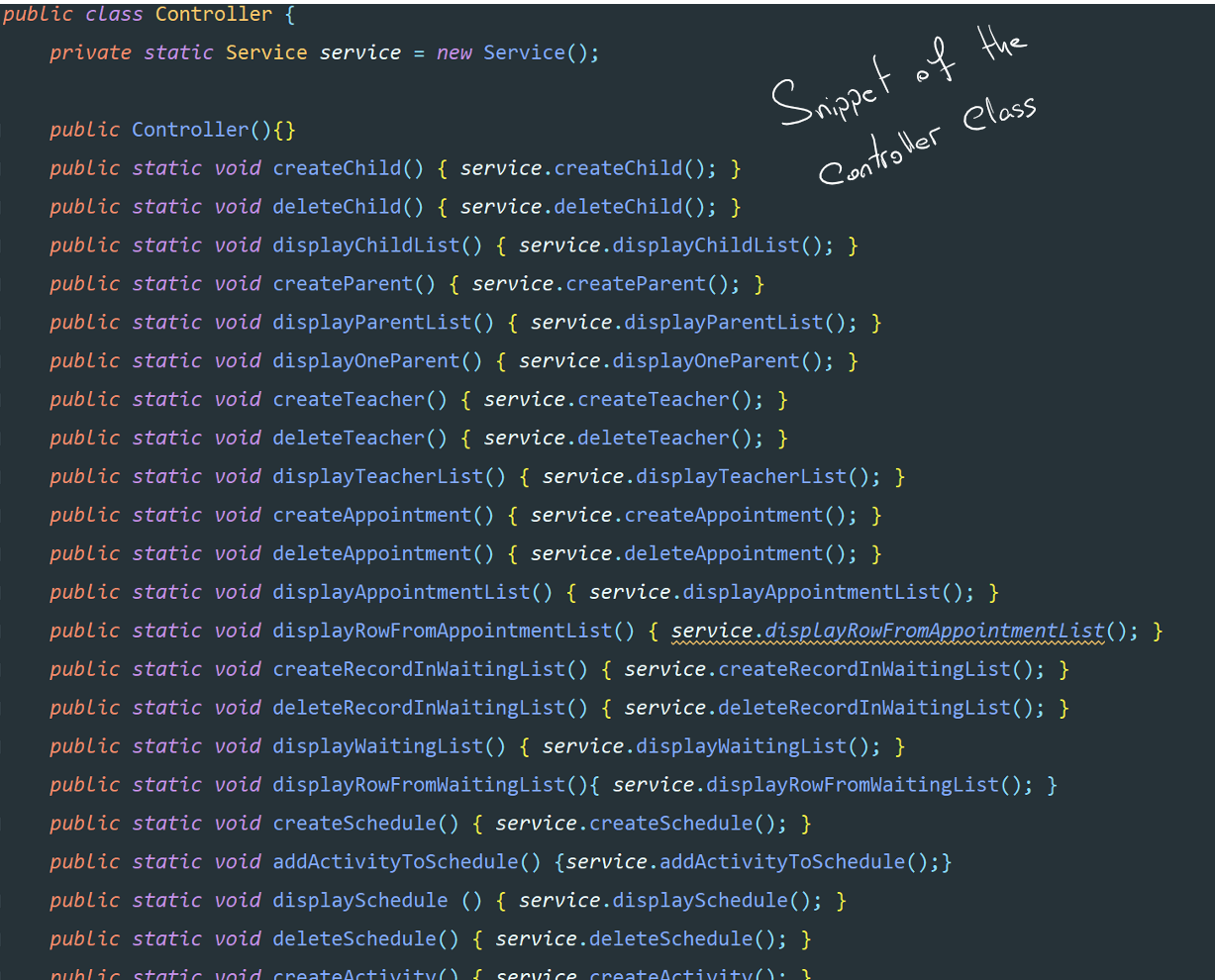
After we had the database all ready to be used, we created a User Management System so that the DB can properly work with the Login System created in the Java Program ( we have created an admin user & users for all teacher ( also included user creation for every time a teacher is created in the database ) all with proper, secure, passwords seeing as the database can pretty much be accessed by anyone with the right credentials ).

The diagrams & snippets of SQL code can be found in Appendices Section, or [here](#_Database).

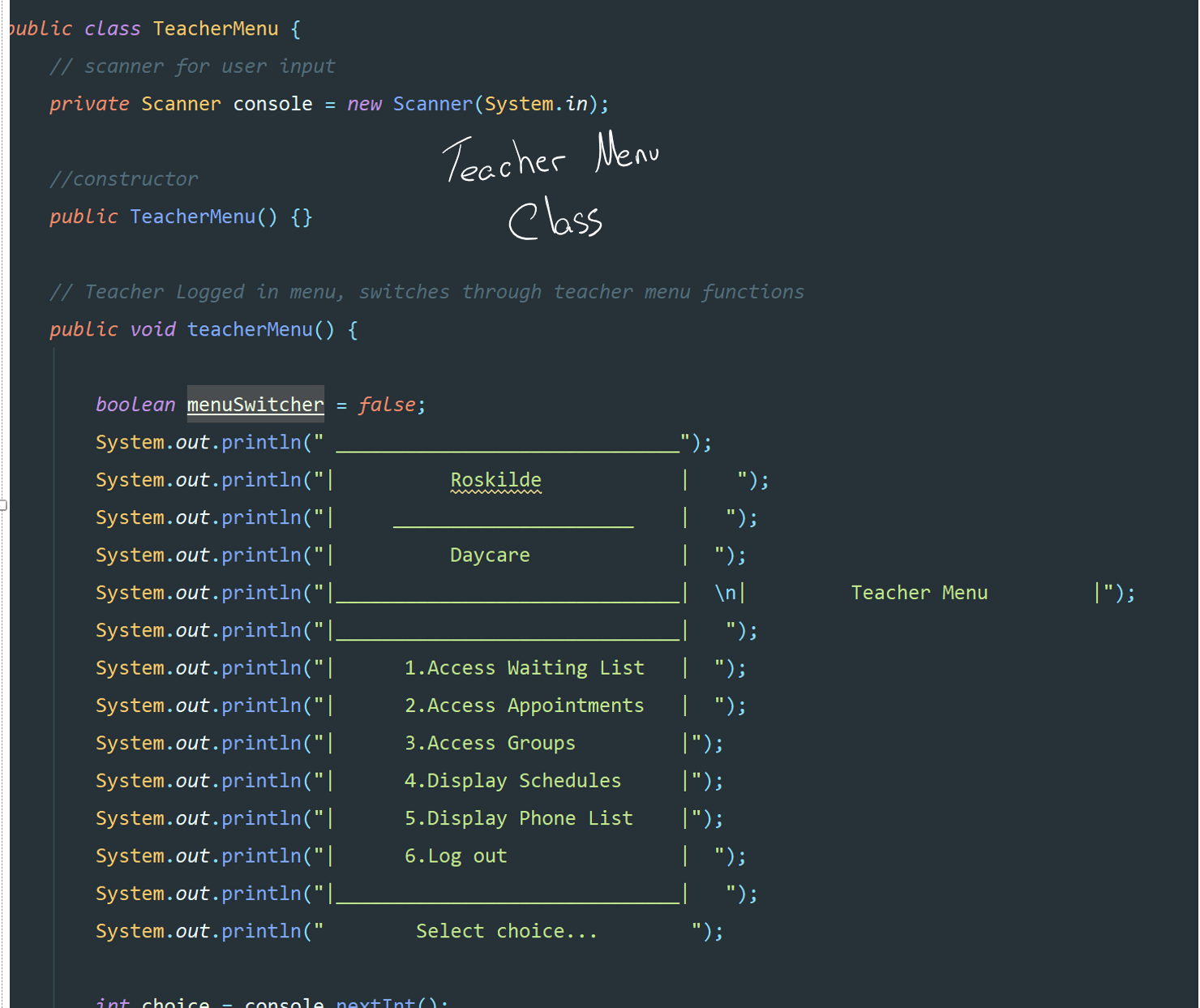
# Java Development

In our program we tried to make our information flow as clear as we could. We designed class menu to display available features to user and send information about needed information to controller. Next, controller class sends information to service class in which we extract information from database and display what needed to the user.









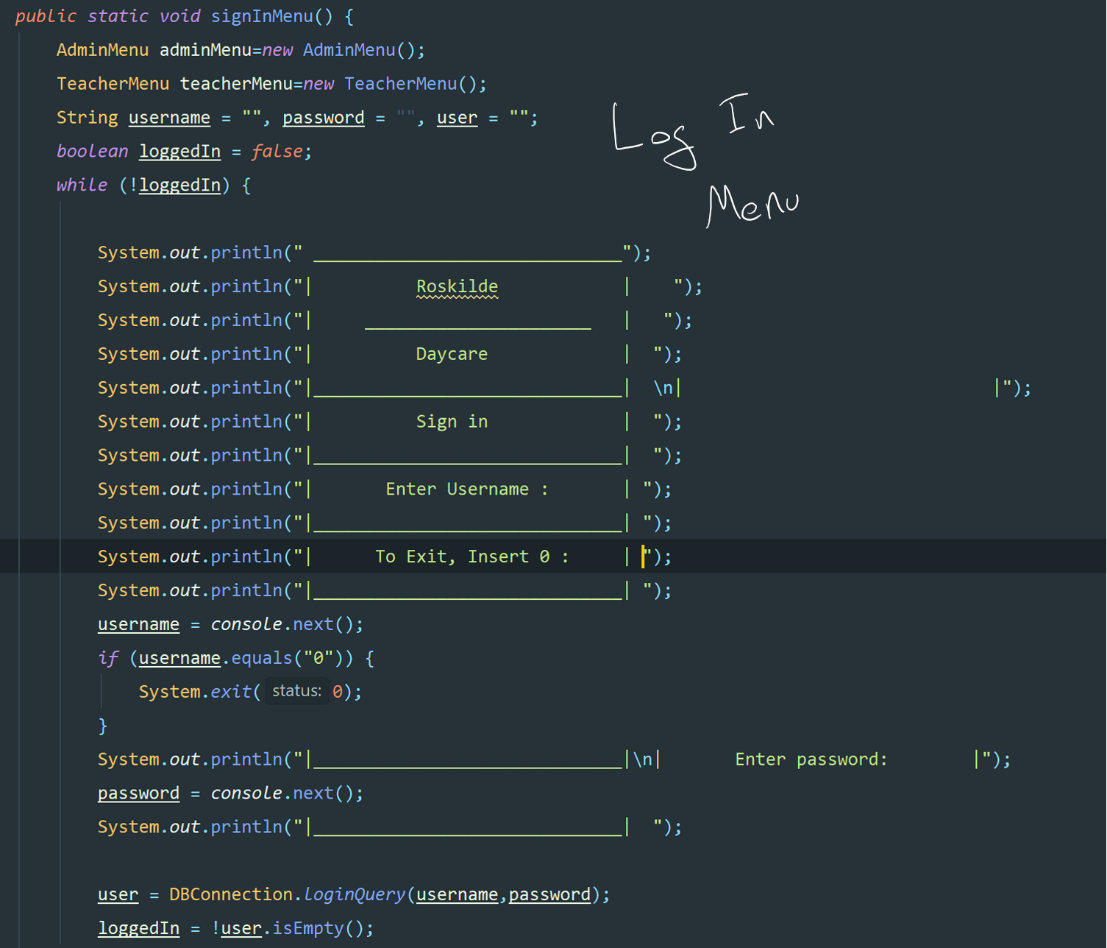
# Snippets Of Code

We have decided to present snippets of code :

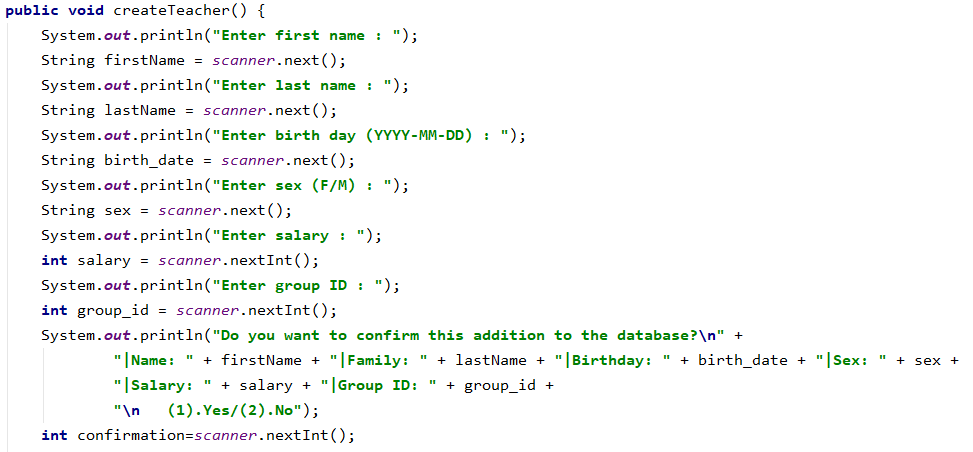
* Log in as administrator
* Create teacher
* Delete child
* Display list of parents

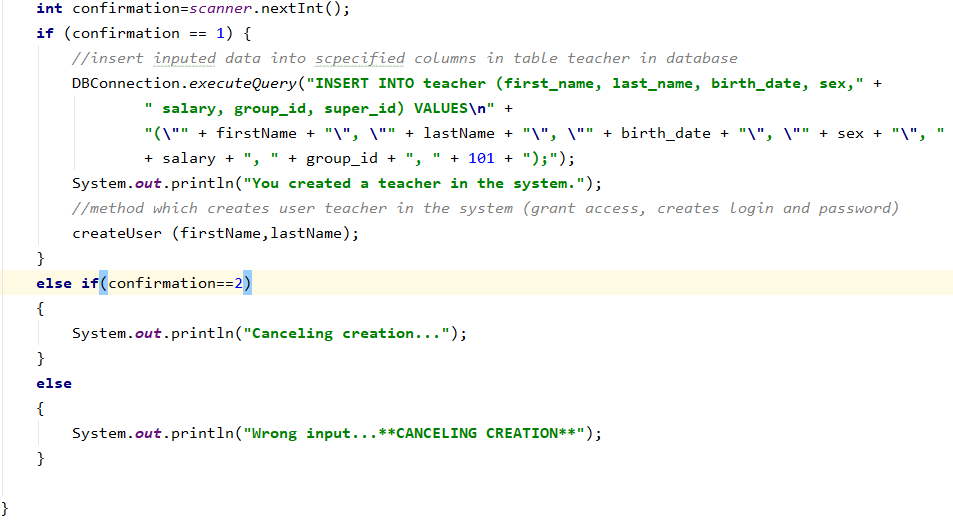
These methods represent general idea how our program works. In order to see the rest of the methods we encourage you to look at our source code attached to the report.

Log in as administrator

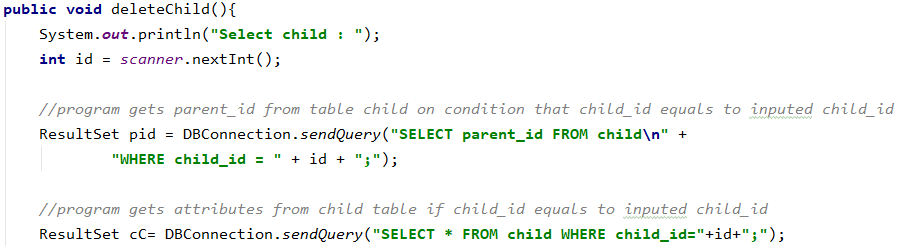


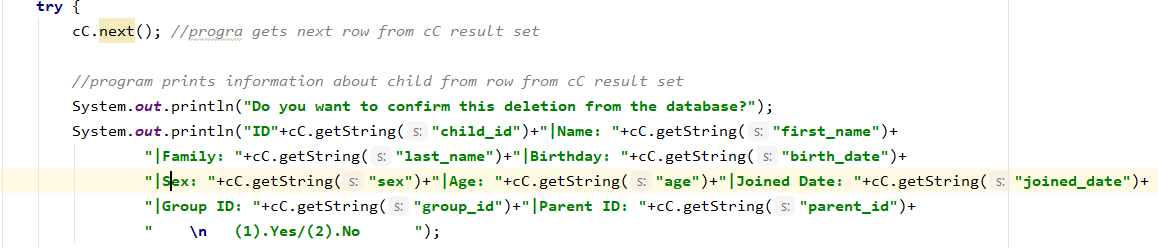
Create teacher method



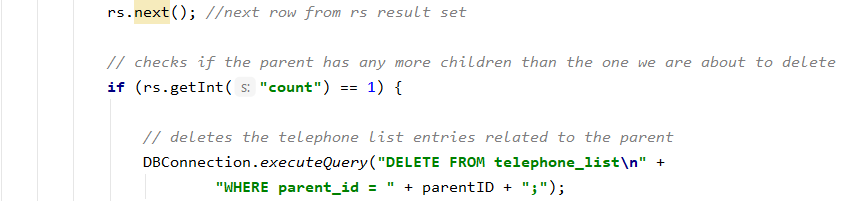


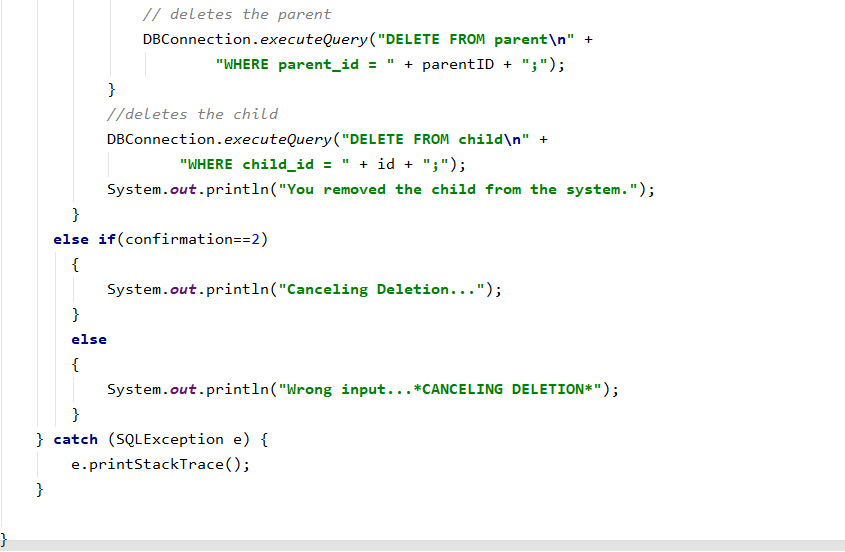
Delete child









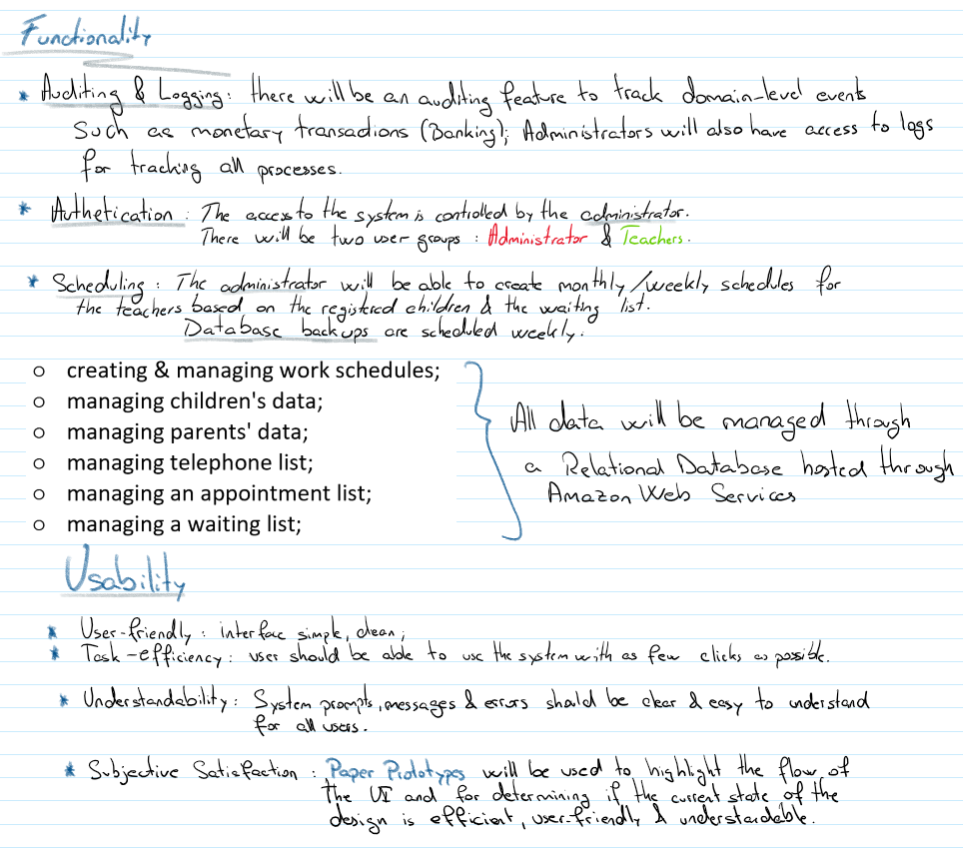


Display list of parents



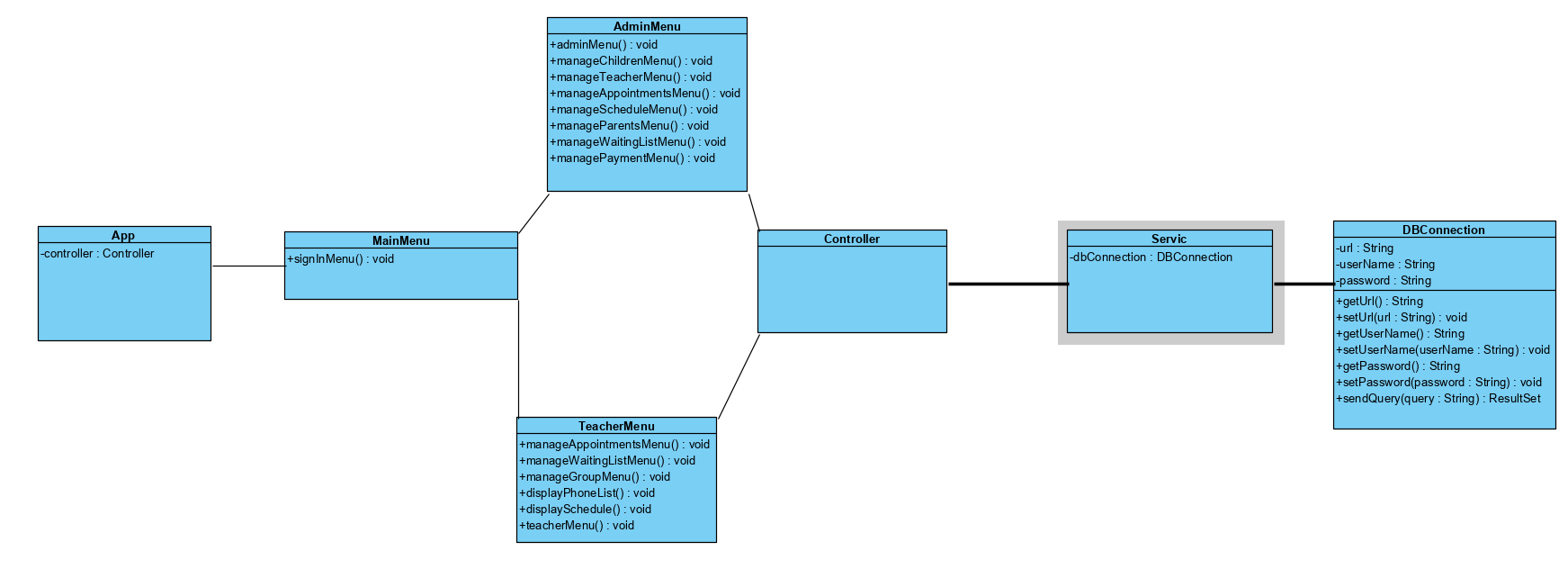
# Appendices

## FURPS+

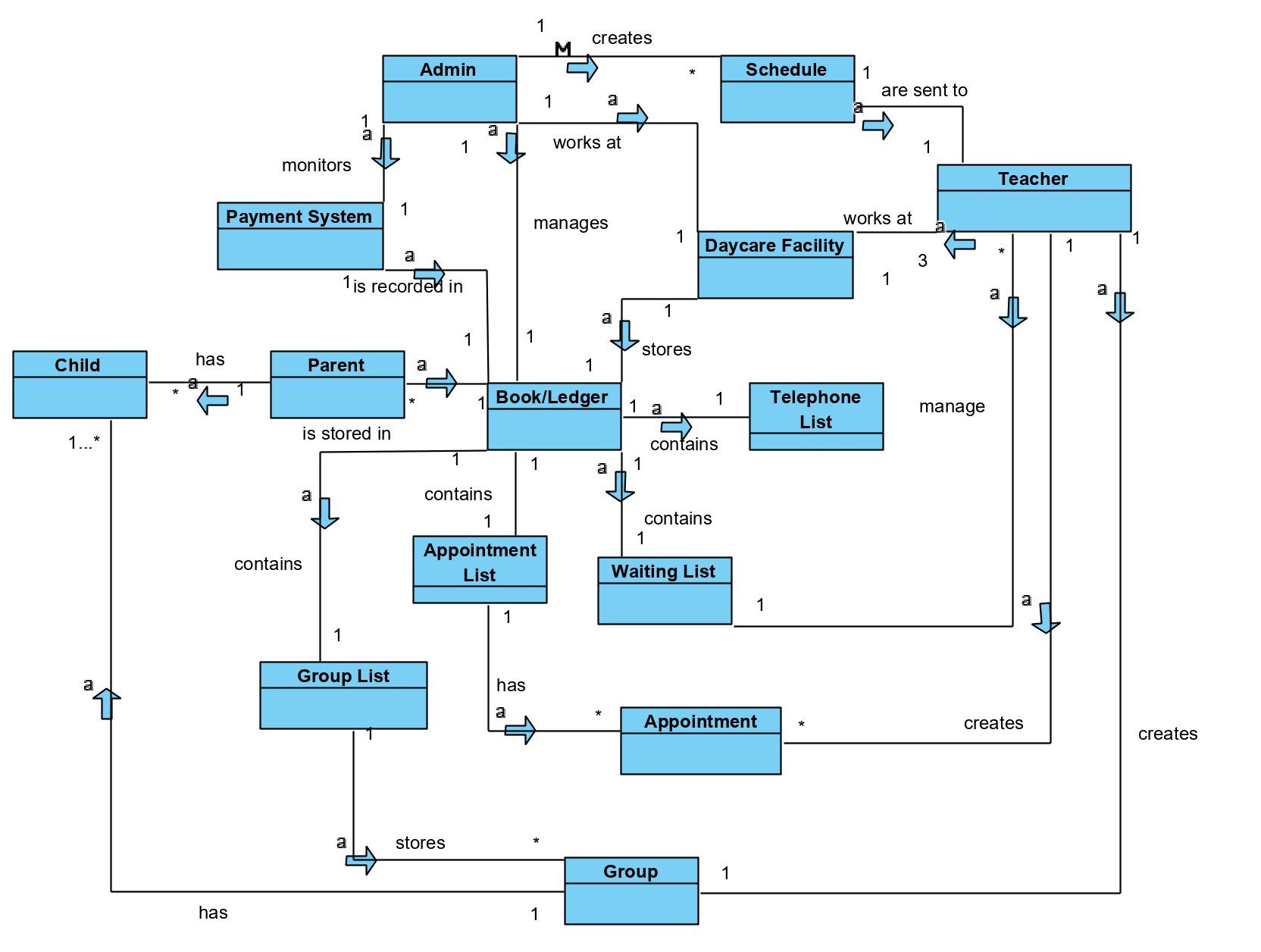


# Diagrams

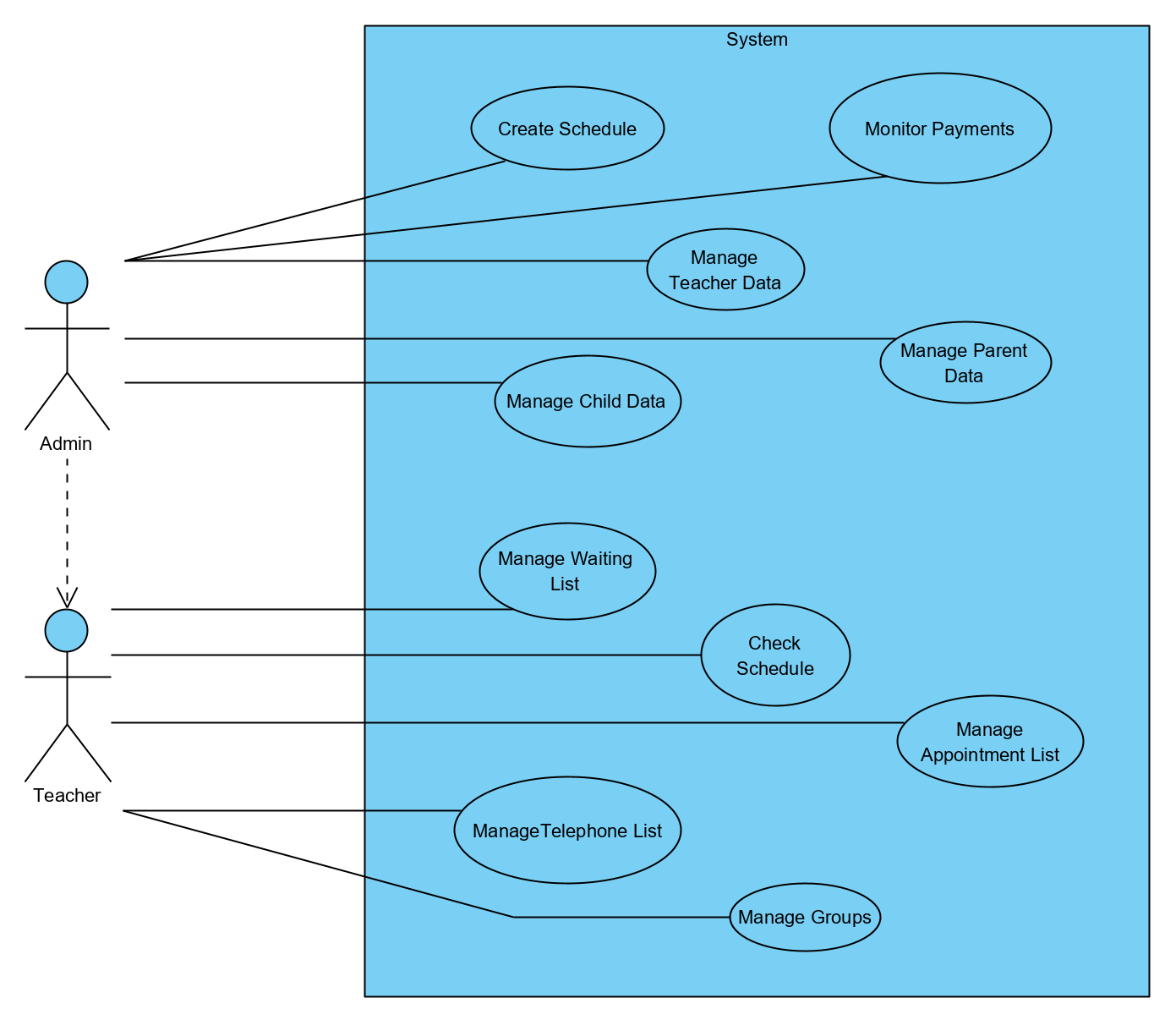
# Class Diagram



# Domain Model

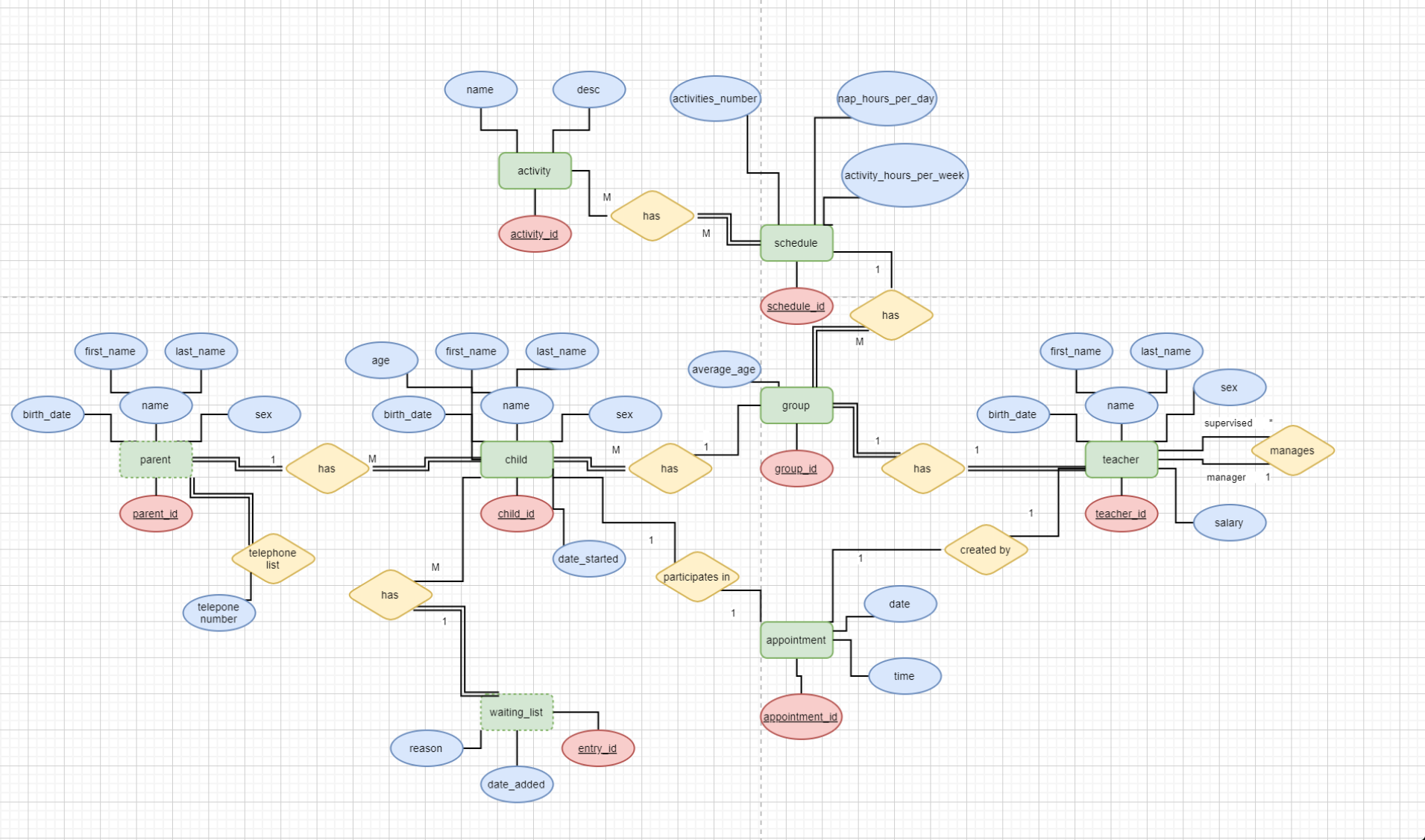


# Use-Case Diagram

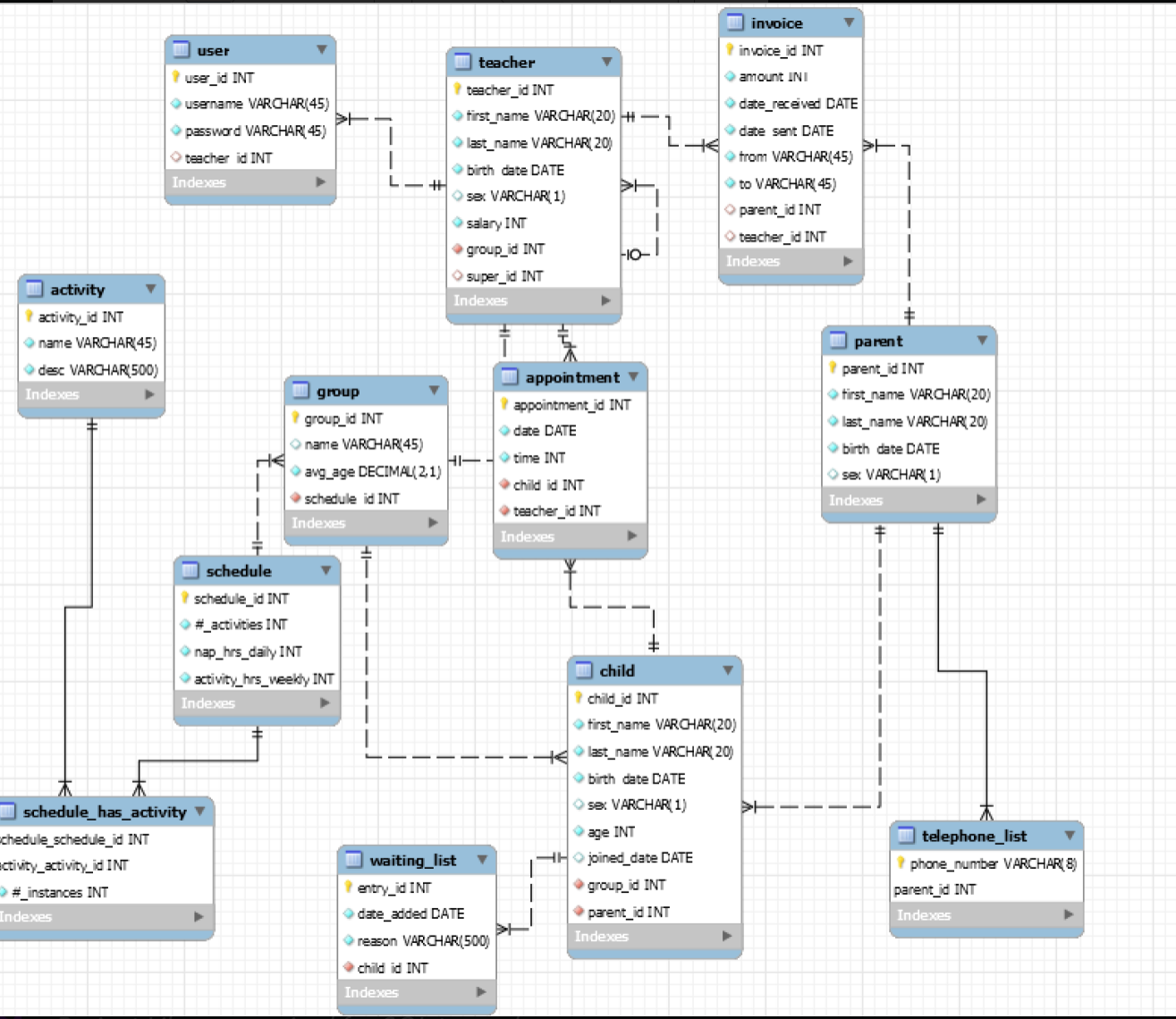


# Database

# ERD



# EERD



# SQL Snippets







