CPSC 471: DATA BASE MANAGEMENT SYSTEMS

Ahmed Hasan

Erin Danielle Paslawski

Amir Hussain

Daycare Management System

Fall 2020

Database Management Systems

Daycare Management System

Ahmed Hasan, Erin Danielle Paslawski, Amir Hussain

Abstract

Today's society is increasingly trying to digitalize the systems we currently use to make them more accessible and efficient. In a traditional daycare system, using paper records to manage employees, children, and parents and all their associated records is not only time consuming, but also error prone and hard to share amongst other people. As well, parents want to have more decisions and control of their children's care and education. In response to this issue, we developed a daycare database system that keeps track of all the data needed in the daycare and allows it to be used to make the daycare more streamlined and efficient. The data can be updated and viewed by parents, admins, and caretakers. To implement this, we created a MySQL relational database, with a RESTful API to query, update and delete data in the database. The database is designed so the different sets of data can be combined in many ways to answer different questions a user may have. The API was written using PHP, and calls stored procedures in the database. The usefulness of creating this API is the ability to easily add a front-end application, making it easier to access and edit data in the database for all users. The system we developed in this project would greatly reduce the need for an admin at a daycare to be spending time on manual record keeping, and it allows the parents access to their records at all times. The parents can then select specific caretakers for their child, add special needs their child may have, and access reports for their child. This will not only increase the efficiency of the daycare but increase the ability for the daycare system to expand and share information with parties outside the administration.

Table of Contents

1 Introduction

It is difficult to find any program that has a combination of a daycare and a nanny. The problem we are addressing with our project is the inability to personalize a child's needs in a daycare system. Specifically, the inability to combine a nanny with a daycare program. Our problem arises when a parent or guardian is registering their child and there is no option to select a nanny that can provide better one on one support for the child. Our solution allows parents to meet their child's as well as their own needs when enrolling their kid. This is why we implemented a database that tracks that information, so it is easily accessible to every parent and the caretaker. Our goal was to give easy access to parents and staff to information they need when taking care of children. The rest of the report will go into further details describing the problem definition, problem solution, project design and implementation of the design.

1.1 Problem Definition

Throughout history in many countries, education was not considered a priority, rather, it was a luxury. Different families educated their young children in different ways. However, in recent years, research has emphasized the importance of early childhood development (Britto, 2017) increasing the need for childcare as more parents are busy working. While some parents hire nannies or babysitters to take care of children and act as a tutor until school age, others may send their kid to a daycare center.

Every parent has their own preference and there are countless reasons as to why they would prefer one over the other. For instance, having a nanny means that they are focused on the child's individual needs. There is no competition for attention and the nanny can solely concentrate on the child's development. There is also a huge safety concern because when hiring a nanny, you can do some research and hire someone you can feel comfortable with and trust your child with. On the contrary, daycares provide the benefit of socialization where kids can learn and grow with other children. The social and intellectual development of children in daycare are stimulated by other kids and their daily experiences (Shin, 2019). Oftentimes children with special needs do not get the same treatment as a normal child. For example, a child with a disability may require extra attention. Though the parents will want to place their child in an institution where they are attended to, the parent would consider social interaction with other children for their child's development. This becomes a dilemma because one staff member may not be able to always take care of the special needs child while allowing for interactions with other children.

Nowadays, what you will not find anywhere is a hybrid; an in-between of a nanny and an institutional daycare program. It is difficult to meet the specific requirements of each parent and the individual needs of the child. When someone registers their kid to a daycare, there is not a lot of customization that can be done. The parent just picks the daycare, and everything else is taken care of by the daycare staff. The child is assigned a class where they follow a schedule given to them by their caretaker. Today, if a parent wants to assign a specific nanny to their kid and still wants to expose their kid to an environment outside their home, then currently there is no known program that has that kind of flexibility. A system that allows the parents or guardians to choose preferences for their children is essential at childcare centers.

Furthermore, there is also the issue of records management. Keeping track of all children's specific medical conditions, treatments, and schedules is overwhelming. There is no reliable system that can provide this type of records management. There is a need for digitizing childcare information. It gives an easy communication between the caretaker and the parent, allowing them both the ability to get and store vital information easily.

1.2 Problem Solution

To address the problems around records management in a daycare setting, as well as the issue of parents not being able to communicate vital information clearly and concisely to the caretaker, we created a daycare system database and associated REST API to query the data. The MySQL database uses multiple tables to store relational information and can be queried to provide the data needed. The information in the tables can also be updated, deleted, or entered by the parents, caretakers and administrators. We built stored procedures in the database and created endpoints to access or edit information through the stored procedures. The system can currently be expanded by adding a user interface for easier use of the API.

2 Project Design

2.1 System Users

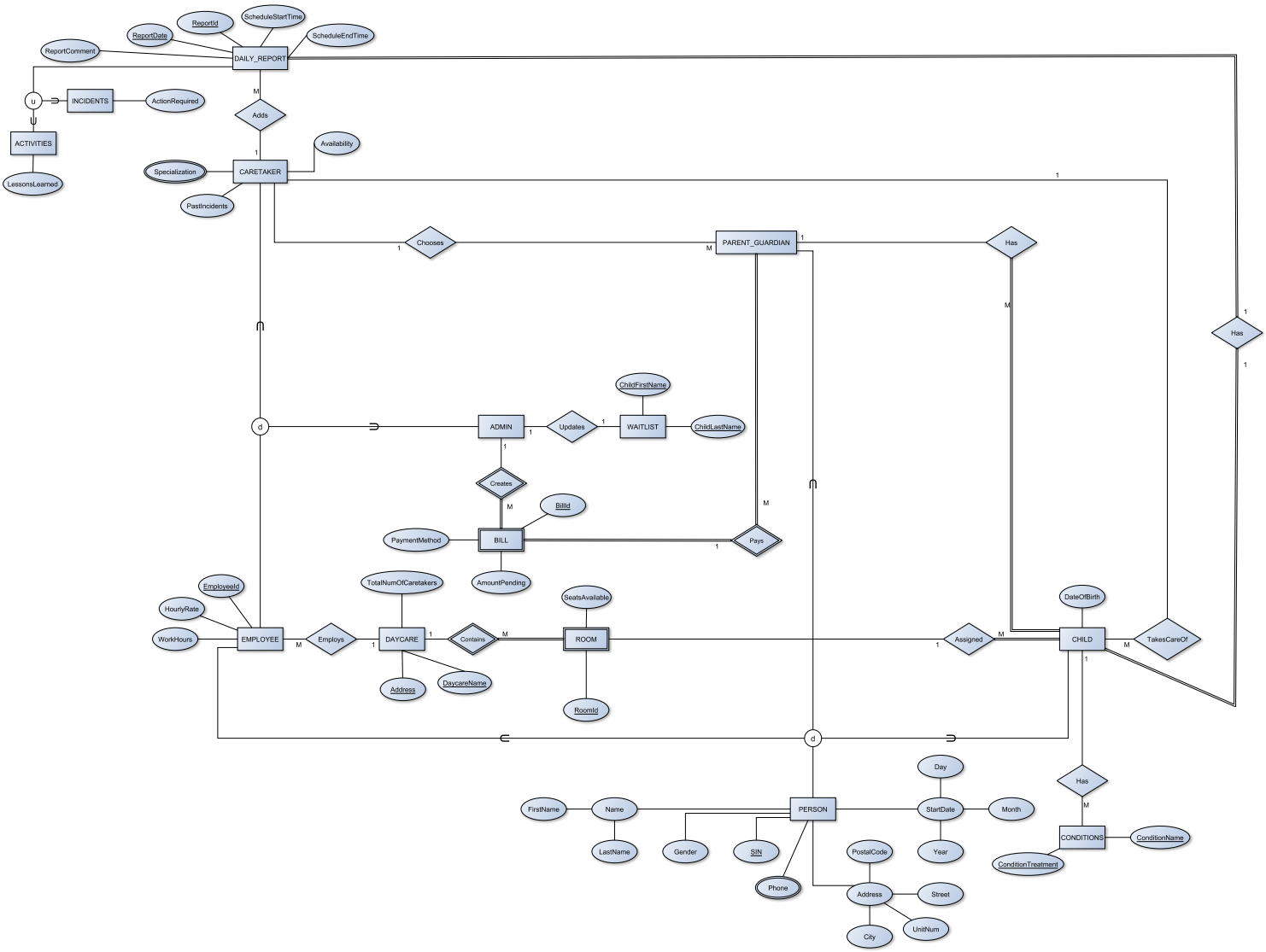
The API allows three primary users to login and interact with the system. The three user roles for the application are the daycare administrators, the children caretakers, and the parents or guardians of the children. Each user has their own access with different privileges.

Admin: An administrator is an employee of the daycare with more privileges than the other users of the daycare system. Since an admin has the highest privileges, they can make all the API calls in our daycare system. The main purpose of the admin is to manage simple administrative daycare operations such as adding or removing employees, updating costs for childcare, and adding children to the waitlist. The admin can manage staff information such as their work hours and their salary information. If a child needs to be added to the waitlist, this would need to be done by the daycare admin. Another task of the admin users is to create and update bills for parents. Once they create a bill and add an outstanding payment to it, the admin can assign it to the parent. When an employee leaves the daycare, the admin also has privileges to delete that employee's records if the daycare decides not to keep it.

Caretaker: A caretaker is also an employee who takes care of the children. They have a medium level authorization; they do not have the authorization to complete administration level tasks. When a parent signs up their child, they can do a little research on the difference caretakers available and view information like any specializations the caretakers have or any incidents that occurred with a caretaker. When the parent is registering their child, they then assign a caretaker that is available. Also at the time of registration, the parent can add any special conditions or restrictions that a child may have. All this information is then available for the caretaker that was assigned the child. By having this information about the child, the caretaker can make appropriate decisions regarding the child and be there for full support. Every day, the caretakers are tasked to write a daily report for the children they are assigned so the parents can have a quick update for their child. Since a caretaker would be working and spending most of the time with the children, the caretakers have privileges to create and get the daily reports for each child.

Parent/Guardian: A parent or guardian has children attending the daycare. A parent has the least number of privileges. Before enrolling their children, parents can browse the daycares and the employees. They can view any information regarding the caretakers that will help them choose an appropriate nanny that will meet the individual needs of their child. Near the end of the day, the caretakers create a daily report for the children and once they make an entry, the parents can log in and view their child's daily report. This daily report is a summary of the child's day and any incidents that might have occurred with the child. Another task that the parent or guardian can do is view their bill from the daycare. This bill is managed by the admin and is updated accordingly. The parents can view any outstanding bill and when they are ready, they can pay with a method of their choice.

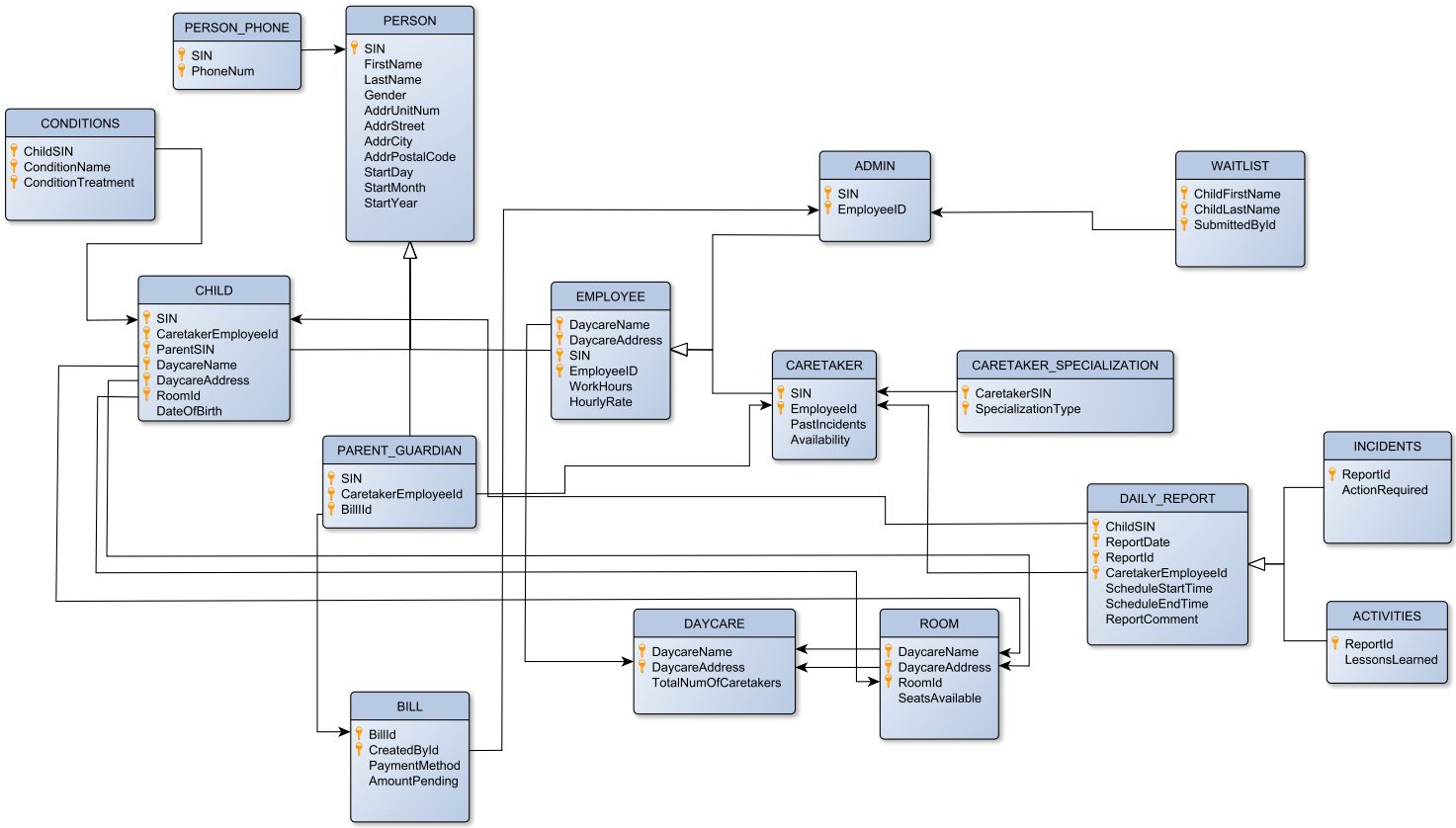
2.2 Extended Entity Relationship Diagram



3 Implementation of the Project Design

3.1 Relational Model Diagram

The relational model outlines all the entities, their attributes, and the relationships we used to build the daycare database. There was a specific algorithm that was used to convert our EERD to a RM. This will be discussed more in section 3.2: EERD to Relational Schema.



3.2 EERD to Relational Schema

From the Extended Entity Relationship Diagram shown in section 2.2, we created a relational model that helped us implement the different tables required in our database. From the EERD diagram, we took each entity and created a table for the entity. We then added the attributes of each entity to its table. In cases of inheritance, we created a base table (such as Person) that contains all the overlapping attributes of the child entities, and then created separate tables for each child entity that would contain the specific attributes, as well as an attribute to link the child table back to the parent. In the case of the Person table, we used the primary key of SIN to link the child tables of Employee, Child, and Parent_Guardian back to Person. This is how we were able to have a separation between the specialized cases of Person. We used a similar approach to the Daily_Report table, and its child tables, Incidents and Activities, which are related to each other through the ReportId.

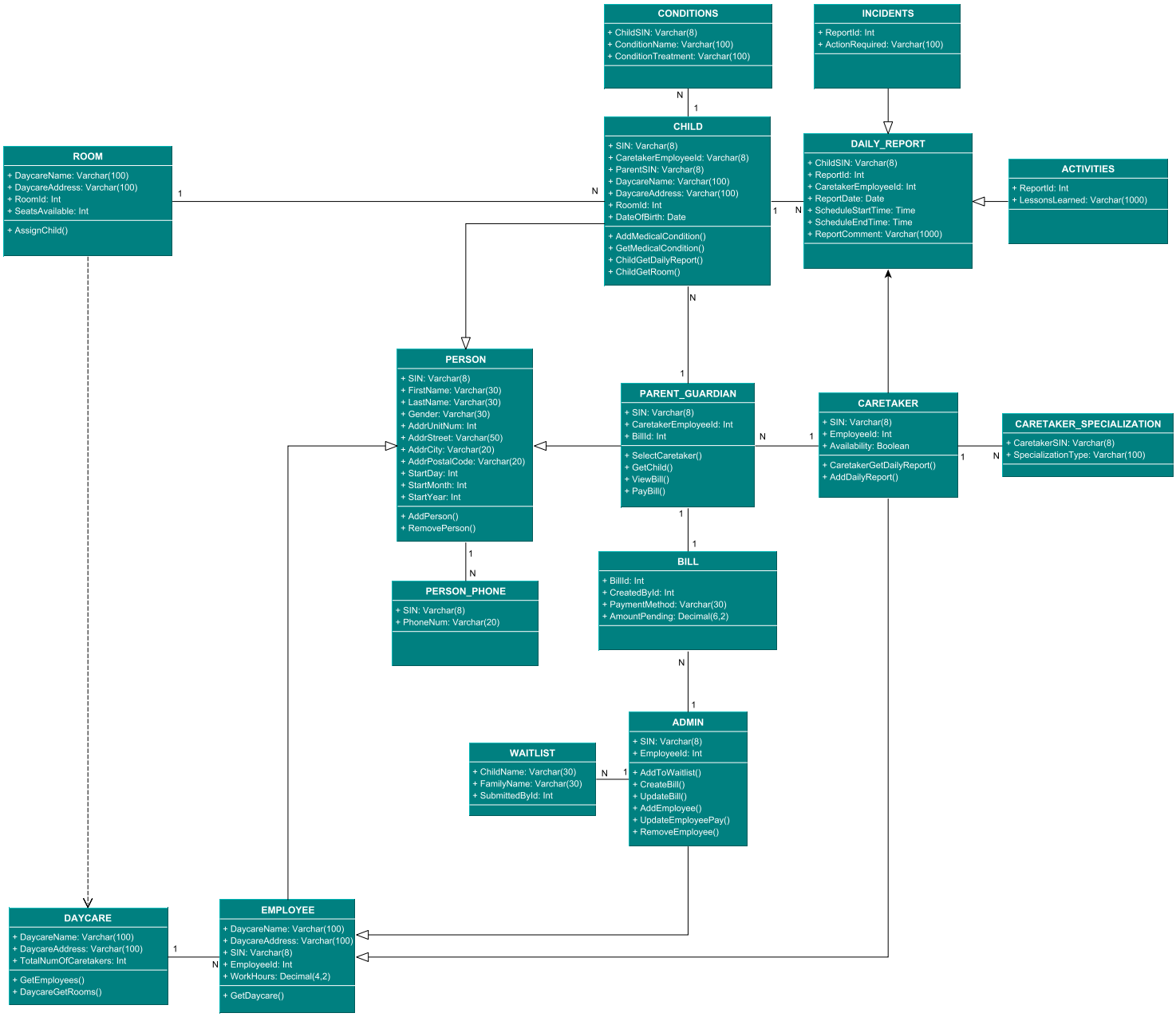
When converting from the EERD diagram to the relational model, we had to make some decisions on the logic or relationships of our entities. For example, we changed the relationship between Caretaker and Caretaker_Specialization from 1:1 to 1:M. This made sense to us, as we created a separate table for the specializations, and linked it back to the Caretaker based on the SIN of the caretaker. In this way we can represent caretakers that have multiple specializations without having duplicates in the Caretaker table.

Another example of an unusual decision we had to make when converting from the EERD to the RM was how to link small, specific tables to the general model. A good example of this is the Waitlist entity. In the EERD, it has two attributes: ChildFirstName and ChildLastName. When we converted to the RM, we added a third attribute of SubmittedById to relate the Waitlist table to the Admin table. Although it may not inherently be necessary to keep track of who entered the waitlist additions, it helps us to be able to relate the tables to each other in our RM.

To deal with differing relationships, we added multiple primary keys to make sure entries in the tables were unique. For example, as shown in the EERD diagram, a Parent/Guardian can have multiple children, but a child can only have one Parent/Guardian. To implement this in our RM, we did not add any attributes from Child in the Parent_Guardian table, but the Child table has the primary key of SIN, CaretakerEmployeeId, ParentSIN, RoomId. By doing this, we can't have more than one instance in the Child table with the same SIN and ParentId, but because the ChildSIN is not an attribute in Parent_Guardian, any number of children can be linked to the same Parent (as long as they have different SINs).

3.3 Object-Oriented Model

To help with the implementation of the API, we made an OOM. From this we were able to decide on the table attribute types and the different endpoints we would need to implement. The SQL queries for each function can be found in Appendix A.



3.4 Database Management System

We used MySQL, a popular open-source database management system. It provided us with a structured collection of our daycare data. The reason we chose to host our data with MySQL is because MySQL is very popular, stable and has a large community for all kinds of support. MySQL database is a relational database, so we created different tables to store all our data. Our database that we created is called childDaycare. To create and manage our MySQL database, we used a graphical user interface called phpMyAdmin. The different tables that we created to store and manage our daycare data are: activities, admin, bill, caretaker, caretaker_specialization, child, conditions, daily_report, daycare, employee, parent, person_phone, room, and waitlist. Refer to Appendix B for a description of each table and to view screenshots of sample data.

We defined the different functionalities as stored procedures in our daycare database. The API did not deal with any plain queries, instead they executed the stored procedures. When we call those specific endpoints, it performs those SQL commands. These stored procedures were used to perform CRUD operations such as selecting data from different tables and updating tables. Stored procedures also allowed for reusability in the event we needed to call several statements in one endpoint we can do that easily and efficiently. To ensure the integrity of our database we also implemented input sanitization to combat things such as SQL injection attacks. SQL injection is possible when you do not check what input is coming in. This means a user can for example drop an entire database if the right command is entered in. We combat this by checking input and stripping special html characters to ensure that this is not possible. That way if someone tries to enter in any malicious content, it will strip away the necessary characters and fail to execute.

4 API Documentation

There is a professional API documentation available, built using Postman's API documentation tool. The documentation shows a description of each endpoint along with the type and expected parameters. Every endpoint has some examples of successful and failed sample requests and shows the different outcomes possible.

API Documentation: `https://documenter.getpostman.com/view/13545898/TVmLDe7R`

5 Deployment

In order to run our project, first get a web server and then install PHP and MySQL. If you have a Windows operating system, it is recommended that you install AppServ which is an Apache distribution containing MySQL and PHP. For MacOS, you can install XAMPP or MAMP. Secondly, it is needed to download the source files from the GitHub repository. Finally, to test the endpoints you will need an API testing tool such as Postman.

The project contains a README file which has further instructions on how to set up the database. To quickly test and run the project on your machine, you can simply follow the "Quick Set Up" steps. Otherwise, there is a more advanced guide to change the database configuration such as database name, server password, etc. by following the "Changing The Database Configuration" steps.

Once you run the script to create the database, tables, and stored procedures, you can start making API calls using the API testing tool.

GitHub Repository: `https://github.com/AMax23/CPSC471-DatabaseProject`

6 Conclusion

To conclude, we are comfortable saying that the project was mostly successful. We implemented the majority of our functionality, including all the core functionality, and we learned a massive deal about database management systems and creating REST APIs with PHP.

7 References

[1] Britto, P. R., Lye, S. J., Proulx, K., Yousafzai, A. K., Matthews, S. G., Vaivada, T., ... & MacMillan, H. (2017). Nurturing care: promoting early childhood development. The Lancet, 389(10064), 91-102.

[2] Shin, M. (2019). Exploring multisensory experiences in infants' learning and development in the child care classrooms. Early Child Development and Care, 1-12.

[3] Elmasri, R., & Navathe, S. (2007). Fundamentals of database systems. Boston: Pearson/Addison Wesley.

[4] D. (2020, November 27). Create Simple PHP 7: 8 CRUD REST API with MySQL & PHP PDO. Retrieved November 23, 2020, from https://www.positronx.io/create-simple-php-crud-rest-api-with-mysql-php-pdo/

Appendices

Appendix A SQL Queries For Stored Procedures

```
/* Table: CHILD
 * Method: AddMedicalCondition()
 * Description: Gets all the conditions and treatments a child may have
 * @param chldSIN - SIN of Child
 * @param cndtnName - The name of the medical condition
 * @param cndtnTrtmnt - What is used to treat the condition
 */
INSERT INTO CONDITIONS (ChildSIN, ConditionName, ConditionTreatment)
VALUES (chldSIN, cndtnName, cndtnTrtmnt);
```

```
/* Table: CHILD
 * Method: GetMedicalCondition()
 * Description: Gets all the conditions and treatments a child may have
 * @param childSIN - SIN of Child
 */
SELECT
      chld.SIN
    , prsn.FirstName
    , prsn.LastName
    , cndtn.ConditionName
    , cndtn.ConditionTreatment
FROM CHILD as chld
INNER JOIN
    CONDITIONS as cndtn
    ON cndtn.ChildSIN = chld.SIN
INNER JOIN
    PERSON as prsn
    ON prsn.SIN = chld.SIN
WHERE chld.SIN = childSIN;
```

```
/* Table: CHILD
 * Method: ChildGetRoom()
 * Description: Get the room for the child
 * @param childSIN - SIN of Child
 */
SELECT
      chld.SIN
    , chld.RoomId
    , rm.DaycareName
    , rm.DaycareAddress
FROM CHILD as chld
INNER JOIN
      ROOM as rm
      ON rm.DaycareName = chld.DaycareName
          AND rm.DaycareAddress = chld.DaycareAddress
          AND chld.RoomId = rm.RoomId
WHERE chld.SIN = childSIN;
```

```sql
/* Table: CHILD
 * Method: ChildGetDailyReport()
 * Description: Gets all daily reports for the child ordered by date (desc)
 * @param childSIN - Child SIN
 */
SELECT
      chld.SIN
    , prsn.FirstName
    , prsn.LastName
    , dlyRprt.ReportId
    , dlyRprt.CaretakerEmployeeId
    , dlyRprt.ReportDate
    , dlyRprt.ScheduleStartTime
    , dlyRprt.ScheduleEndTime
    , dlyRprt.ReportComment
    , actvts.LessonsLearned
    , incdnts.ActionRequired
FROM DAILY_REPORT as dlyRprt
INNER JOIN
      ACTIVITIES as actvts
      ON actvts.ReportId = dlyRprt.ReportId
INNER JOIN
      INCIDENTS as incdnts
      ON incdnts.ReportId = dlyRprt.ReportId
INNER JOIN
     CHILD as chld
     ON chld.SIN = dlyRprt.ChildSIN
INNER JOIN
     PERSON as prsn
     ON prsn.SIN = chld.SIN
WHERE chld.SIN = childSIN
ORDER BY
     dlyRprt.ReportDate DESC
   , dlyRprt.ScheduleStartTime DESC;
```

```sql
/* Table: PARENT_GUARDIAN
 * Method: SelectCaretaker()
 * Description: View all caretakers and their info the specified daycare
 * @param dycrName - Name of the daycare
 * @param dycrAddr - The address of the daycare
 */
SELECT
      prsn.FirstName
    , prsn.LastName
    , prsn.Gender
    , emp.DaycareName
    , emp.DaycareAddress
    , prsn.StartDay
    , prsn.StartMonth
    , prsn.StartYear
    , crtkr.PastIncidents
    , crtkr.Availability
    , crtkrSpclztn.SpecializationType
FROM CARETAKER as crtkr
INNER JOIN
      PERSON as prsn
      ON prsn.SIN = crtkr.SIN
INNER JOIN
      CARETAKER_SPECIALIZATION as crtkrSpclztn
      ON crtkrSpclztn.CaretakerSIN = crtkr.SIN
INNER JOIN
      EMPLOYEE as emp
      ON emp.EmployeeId = crtkr.EmployeeId
WHERE emp.DaycareName = dycrName
     AND emp.DaycareAddress = dycrAddr;
```

```
/* Table: PARENT_GUARDIAN
 * Method: GetChild()
 * Description: Returns child(ren) of parents
 * @param prntSIN
 */
SELECT
      chld.SIN
    , prsn.FirstName
    , prsn.LastName
    , chld.DateOfBirth
FROM CHILD as chld
INNER JOIN
      PERSON as prsn
      ON prsn.SIN = chld.SIN
WHERE chld.ParentSIN = prntSIN;
```

```
/* Table: PARENT_GUARDIAN
 * Method: ViewBill()
 * Description: View outstanding payment due.
 * @param prntSIN - The SIN of parent
 */
SELECT
      prnt.SIN
    , bill.AmountPending
FROM PARENT_GUARDIAN as prnt
INNER JOIN
      BILL as bill
      ON bill.BillId = prnt.BillId
WHERE prnt.SIN = prntSIN;
```

```
/* Table: PARENT_GUARDIAN
 * Method: PayBill()
 * Description: Parent pays outstanding bill
 * @param id - id of the bill to be paid
 * @param pmntMthd - payment method of parent_guardian
 * @param amntPndg - the amount left over after payment
 */
UPDATE BILL
SET PaymentMethod = pmntMthd,
    AmountPending = amntPndg
WHERE BillId = id;
```

```
/* Table: ADMIN
 * Method: AddToWaitlist()
 * Description: Add a new Child/family to the waitlist
 * @param chldFrstNme - Name of child to be inserted
 * @param chldLstNme - Last name of child/family
 * @param empId - ID of employee submitting
 */
INSERT INTO WAITLIST (ChildFirstName, ChildLastName, SubmittedById)
VALUES (chldFrstNme, chldLstNme, empId);
```

```
/* Table: ADMIN
 * Method: CreateBill()
 * Description: Add a new bill
 * @param bill - Id of bill
 * @param empId - ID of employee submitting
 * @param method - Payment method
 * @param amount - amount of bill
 */
 INSERT INTO BILL (BillId, CreatedById, PaymentMethod, AmountPending)
 VALUES (bill, empId, method, amount);;
```

```
/* Table: ADMIN
 * Method: UpdateBill()
 * Description: Update child bill
 * @param bill - Id of bill
 * @param amount - new amount of bill
 */
 UPDATE BILL
 SET AmountPending = amount
 WHERE BillId = bill;
```

```
/* Table: ADMIN
 * Method: AddEmployee()
 * Description: Add a new employee
 * @param daycare - Daycare Name
 * @param address - Daycare Address
 * @param empSIN - SIN of new EmployeeID
 * @param empId - Employee ID
 * @param wrkHrs - Hours available to work
 * @param hrlyRate - hourly rate of the employee
 */
 INSERT INTO EMPLOYEE (DaycareName, DaycareAddress, SIN, EmployeeId, WorkHours,
                       HourlyRate)
 VALUES (daycare, address, empSIN, empId, wrkHrs, hrlyRate);
```

```
/* Table: ADMIN
 * Method: UpdateEmployeePay()
 * Description: Update employee payroll info
 * @param empSIN - SIN of Employee
 * @param wrkHrs - Hours available to work
 * @param hrlyRate - new Hourly rate of the employee
 */
 UPDATE EMPLOYEE
 SET  WorkHours = wrkHrs
    , HourlyRate = hrlyRate
 WHERE SIN = empSIN;
```

```
/* Table: ADMIN
 * Method: RemoveEmployee()
 * Description: Remove an employee
 * @param empId - ID of employee to remove
 */
 DELETE FROM EMPLOYEE
 WHERE EmployeeId = empId;
```

```
/* Table: PERSON
 * Method: AddPerson()
 * Description: Add a new person
 * @param prsnSIN - SIN of new person
 * @param frstNm - First name
 * @param lstNm - Last name
 * @param gndr - Gender
 * @param untNum - Unit number
 * @param strt - Street address
 * @param cty - City
 * @param pstlCde - Postal code
 * @param strtDy - Start Day
 * @param strtMnth - StartMonth
 * @param strtYr - Start year
 * @param phneNm - Phone number
 */
INSERT INTO PERSON (SIN, FirstName, LastName, Gender, AddrUnitNum, AddrStreet,
                    AddrCity, AddrPostalCode, StartDay, StartMonth, StartYear)
VALUES (prsnSIN, frstNm, lstNm, gndr, untNum, strt, cty, pstlCode, strtDy,
        strtMnth, strtYr);

INSERT INTO PERSON_PHONE (SIN, PhoneNum)
VALUES (prsnSIN, phnNum);
```

```
/* Table: PERSON
 * Method: RemovePerson()
 * Description: Remove a person
 * @param prsnSIN - SIN of person to remove
 */
DELETE FROM PERSON
WHERE SIN = prsnSIN;
```

```
/* Table: EMPLOYEE
 * Method: GetDaycare()
 * Description: Gets the daycare the Employee works at
 * @param empId - Id of employee
 */
SELECT
     DaycareName
   , DaycareAddress
FROM EMPLOYEE
WHERE EmployeeId = empId;
```

```
/* Table: DAYCARE
 * Method: GetEmployees()
 * Description: Gets a list of all employees
 * @param dycreName - the daycarename
 * @param dycreAddress - the address of the daycre
 */
SELECT
     emp.SIN
   , emp.EmployeeId
   , prsn.FirstName
   , prsn.LastName
   , emp.WorkHours
   , emp.HourlyRate
FROM EMPLOYEE as emp
INNER JOIN
     PERSON as prsn
     ON prsn.SIN = emp.SIN
WHERE emp.DaycareName = dycreName
    AND emp.DaycareAddress = dycreAddress;
```

```
/* Table: DAYCARE
 * Method: DaycareGetRooms()
 * Description: Get a list of rooms at the daycare
 * @param dycreName - Name of the daycare
 * @param dycreAddress - the address of the daycare
 */
SELECT
     RoomId
   , SeatsAvailable
FROM ROOM
WHERE DaycareName = dycreName
    AND DaycareAddress = dycreAddress;
```

```
/* Table: CARETAKER
 * Method: AddReport()
 * Description: Insert report into Daily_report table
 * @param chldSIN - SIN of child the report is about
 * @param rptId - Id of the report
 * @param empId - EmployeeID of the caretaker inserting the report
 * @param rptDte - Date of the report
 * @param strtTme - Start time of the schedule
 * @param endtme - End time of the schedule
 * @param rptCmmnt - Text of report
 * @param lsnLrnd - Lessons learned for the day
 * @param actnRqrd - Any action requried from the child
 */
INSERT INTO DAILY_REPORT (ChildSIN, ReportId, CaretakerEmployeeId, ReportDate,
                          ScheduleStartTime, ScheduleEndTime, ReportComment)
VALUES (chldSIN, rptId, empId, rptDte, strtTme, endTme, rptCmmnt);

INSERT INTO ACTIVITIES (ReportId, LessonsLearned)
VALUES (rptID, lsnLrnd);

INSERT INTO INCIDENTS (ReportId, ActionRequired)
VALUES (rptID, actnRqrd);
```

```
/* Table: CARETAKER
 * Method: CaretakerGetDailyReport()
 * Description: Get daily report submitted by that caretaker
 * @param crtkrId - Id of caretaker
 * @param rptdate - Date wanted
 * @param chldSIN - SIN of child repoported on
 */
SELECT
     dlyRprt.ChildSIN
   , dlyRprt.ReportDate
   , dlyRprt.ScheduleStartTime
   , dlyRprt.ScheduleEndTime
   , dlyRprt.ReportComment
   , actvts.LessonsLearned
   , incdnts.ActionRequired
FROM DAILY_REPORT as dlyRprt
INNER JOIN
     ACTIVITIES as actvts
     ON actvts.ReportId = dlyRprt.ReportId
INNER JOIN
     INCIDENTS as incdnts
     ON incdnts.ReportId = dlyRprt.ReportId
WHERE dlyRprt.ChildSIN = chldSIN
    AND dlyRprt.CaretakerEmployeeId = crtkrId
    AND dlyRprt.ReportDate = rptdate
ORDER BY
     dlyRprt.ReportDate DESC
   , dlyRprt.ScheduleStartTime DESC;
```

```
/* Table: ROOM
 * Method: AssignChild()
 * Description: Assign a child to the room
 * @param chldSIN- SIN of CHILD
 * @param rmId - Id of Room being assigned
 */
UPDATE CHILD
SET RoomId = rmId
WHERE SIN = chldSIN;
```

Appendix B Database Tables

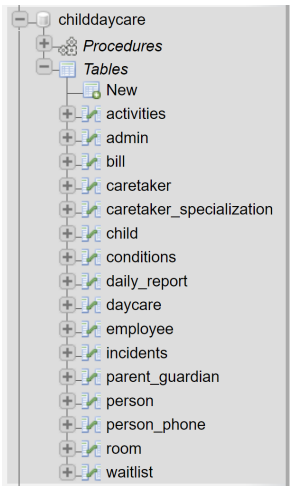


Figure 1: All tables part of the childdaycare

ACTIVITIES: Contains what the child learned for the day. Activities are part of the daily_report.

	ReportId	LessonsLearned
Edit Copy Delete	1	Learned about space and aliens
Edit Copy Delete	2	Learned about dinosaurs
Edit Copy Delete	3	Studied computers
Edit Copy Delete	4	Could not focus on his education
Edit Copy Delete	5	*NULL*
Edit Copy Delete	6	Child can cook!

Figure 2: activities table

ADMIN: An admin is an employee with more privileges than other users of the daycare system, an admin will have authorization over the entire database. They manage simple administrative daycare operations such adding or removing employees and updating costs for childcare.

	SIN	EmployeeId
Edit Copy Delete	99988877	1234

Figure 3: admin table

BILL: Bill contains information on the bill that each parent will have to pay. Stores information on how much there is left to pay, and how they plan on paying.

	BillId	CreatedById	PaymentMethod	AmountPending
Edit Copy Delete	12345	1234	MasterCard	0.00
Edit Copy Delete	43434	1234	*NULL*	30.00

Figure 4: bill table

CARETAKER: A caretaker is an employee who takes care of the children. A caretaker has privileges to create and get the daily reports for each child. A caretaker takes care of children and keeps track of their special restrictions and general behaviour.

	SIN	EmployeeId	PastIncidents	Availability
Edit Copy Delete	12345678	5555	Slept during shift once.	1
Edit Copy Delete	44433222	5550	*NULL*	1

Figure 5: caretaker table

CARETAKER_SPECIALIZATION: Caretakers can specialize in multiple fields that may make them more fit to help certain children more than others.

	CaretakerSIN	SpecializationType
Edit Copy Delete	12345678	Eating Disorders
Edit Copy Delete	12345678	Healthy Eating
Edit Copy Delete	44433222	OCD
Edit Copy Delete	44433222	Trauma

Figure 6: caretaker_specialization table

CHILD: A child is part of a daycare who is taken care of by a caretaker. A child may have special needs that the caretaker would need to pay attention to and deal with appropriately.

	SIN	CaretakerEmployeeId	ParentSIN	DateOfBirth	DaycareName	DaycareAddress	RoomId
Edit Copy Delete	11111111	5555	11122233	2005-10-12	Daycare One	Daycare Street NW	1
Edit Copy Delete	55566777	5550	11111112	2007-06-06	Daycare One	Daycare Street NW	2

Figure 7: child table

CONDITIONS: There are the conditions that a child may have. Caretakers need to know about these special conditions so that they can properly combat them.

	ChildSIN	ConditionName	ConditionTreatment
Edit Copy Delete	11111111	Asthma	Inhaler
Edit Copy Delete	11111111	Celiac Disease	Gluten Free Diet
Edit Copy Delete	55566777	Asthma	Inhaler
Edit Copy Delete	55566777	OCD	Monitor Actions

Figure 8: conditions table

DAILY_REPORT: Children are given a daily report, based on the activities they complete during the day. It is something for the parent to see what the child has done throughout the day.

	ChildSIN	ReportId	CaretakerEmployeeId	ReportDate	ScheduleStartTime	ScheduleEndTime	ReportComment
Edit Copy Delete	11111111	1	5555	2020-01-01	10:00:00	13:00:00	Child took medication at 2pm. All day child did a ...
Edit Copy Delete	11111111	3	5555	2020-07-01	09:00:00	12:00:00	Child played well
Edit Copy Delete	11111111	6	5555	2020-07-01	12:12:00	12:45:00	Child took a lunch break
Edit Copy Delete	11111111	7	5555	2020-07-01	12:45:00	20:00:00	Childs parent was late to pick him up.
Edit Copy Delete	55566777	2	5550	2020-01-01	09:00:00	16:41:00	Child took 20 min breaks as directed. Child was a ...
Edit Copy Delete	55566777	4	5550	2020-07-01	12:00:00	16:41:00	Child was a bad child.
Edit Copy Delete	55566777	5	5550	2020-07-01	15:00:00	20:00:00	Child calmed down after.

Figure 9: daily_report table

DAYCARE: A daycare is a facility which admins, caretakers, parents, and children interact with. Parents have children that may attend the daycare, who is then taken care of by a caretaker. The admins manage the daycare system and the administrative tasks.

	DaycareName	DaycareAddress	TotalNumOfCaretakers
Edit Copy Delete	Daycare One	Daycare Street NW	5
Edit Copy Delete	Daycare Two	Other Daycare Street	30

Figure 10: daycare table

EMPLOYEE: An employee is a worker at the daycare. An employee can either be an admin or a caretaker, and they both have more privileges than a parent.

	DaycareName	DaycareAddress	SIN	EmployeeId	WorkHours	HourlyRate
Edit Copy Delete	Daycare One	Daycare Street NW	12345678	5555	8.00	25.00
Edit Copy Delete	Daycare One	Daycare Street NW	44433222	5550	12.00	30.00
Edit Copy Delete	Daycare One	Daycare Street NW	99988877	1234	10.00	100.00

Figure 11: employee table

INCIDENTS: If there is any incident involving a child, it can be added to the table which will be part of the child's daily report for the parent to view.

	ReportId	ActionRequired
Edit Copy Delete	1	*NULL*
Edit Copy Delete	2	Needs to be talked to about behaviour
Edit Copy Delete	3	*NULL*
Edit Copy Delete	4	This is a bad kiddo
Edit Copy Delete	5	Talked to child about his behaviour
Edit Copy Delete	6	*NULL*
Edit Copy Delete	7	Child was upset that his parent was late

Figure 12: incidents table

PARENT_GUARDIAN: Information about the parent is kept such as their child's caretaker and any bill associated with them.

	SIN	CaretakerEmployeeId	BillId
Edit Copy Delete	11111112	5550	43434
Edit Copy Delete	11122233	5555	12345

Figure 13: parent_guardian table

PERSON: A person is a superclass of employee, parent, and child. They hold the information of general person attributes, things that are common with all users in the database.

	SIN	FirstName	LastName	Gender	AddrUnitNum	AddrStreet	AddrCity	AddrPostalCode	StartDay	StartMonth	StartYear
Edit Copy Delete	11111111	First	Child	Female	10	Home address	Calgary	T2X 2E3	19	7	2011
Edit Copy Delete	11111112	Child	Parent	Male	30	Home address	Calgary	T2X 2E3	9	7	2003
Edit Copy Delete	11122233	Joe	Fresh	Male	456	Also Clagary Dr	Calgary	T2X 2E3	12	10	2008
Edit Copy Delete	12345678	Erin	Employee	Female	123	Calgary Drive	Calgary	T2X 2E3	9	7	2003
Edit Copy Delete	44433222	Dr	Employee	Male	1000	The Mansion	Calgary	T5R 0R4	10	10	2010
Edit Copy Delete	55566777	A Child	Person	Male	10	Also Clagary Dr	Calgary	T5R 0R4	12	10	2008
Edit Copy Delete	99988877	Admin	Lady	Female	107	The Daycare St	Calgary	T5R 0R4	29	10	2008

Figure 14: person table

PERSON_PHONE: Every person may have one or more phone numbers.

	SIN	PhoneNum
Edit Copy Delete	11111112	1123456789
Edit Copy Delete	11122233	4035556543
Edit Copy Delete	12345678	4035551223
Edit Copy Delete	12345678	4035551234
Edit Copy Delete	44433222	4039999911
Edit Copy Delete	99988877	1233334444

Figure 15: person_phone table

ROOM: A room is part of a daycare. Each child is assigned a room.

	DaycareName	DaycareAddress	RoomId	SeatsAvailable
Edit Copy Delete	Daycare One	Daycare Street NW	1	4
Edit Copy Delete	Daycare One	Daycare Street NW	2	1
Edit Copy Delete	Daycare Two	Other Daycare Street	1	8

Figure 16: room table

WAITLIST: The waitlist of children who are waiting to be entered into the daycare. These children are not in the school system yet.

	ChildFirstName	ChildLastName	SubmittedById
Edit Copy Delete	Evil Child	Nice Family	1234
Edit Copy Delete	Possible New Child	Theior Family	1234

Figure 17: waitlist table