

# My Health Patient Portal (MHPP)\*

Carlos Santos

Department of Computer Engineering  
University of Idaho, USA  
Port3116@vandals.uidaho.edu

Keller Lawson

Department of Computer Science  
University of Idaho, USA  
Laws1689@vandals.uidaho.edu

## ABSTRACT

SQL is a formal relational database management language. In this document, we focus on the creation of a My Health Patient Portal project given to us in CS360 (Database Management) at the University of Idaho. The project goal was to design a system from the ground up that uses MySQL as the backbone, a front-end web system, and a middle-end system to query the MySQL database.

## MHPP CONCEPTS

- MySQL; Web-Based *Graphical user interface (GUI)*; *RDMS -> Relation Database Management System*; Faker Data Generation Tool

## KEYWORDS

SQL; PHP; MySQL; JavaScript; HTML; Database; Faker; Python;

---

## 1 INTRODUCTION

The scope of this paper is to describe the major components of our My Health Patient Portal. From here on out MHPP will refer to My Health Patient Portal. The MHPP project has three major pieces to it. The front end or what is referred to as the user interface. This is talked about in section 3. It also has a back end or typically referred to as the RDMS which stands for Relation Database Management System. The RDMS is mentioned in section 1.1, 2, and 5. Finally, we have a third component which is test data generation in section 4.

### 1.1 RDMS using MySQL

For our RDMS, we are using MySQL to host our main schema myhealth2 which contains all of our tables. MySQL is one of the most popular open-source SQL databases and because of this has a lot of forum and community support to assist us when creating our database. While MySQL can lack in speed when scaled to massive databases, it was the proper choice to support our project in its current state.

### 1.2 Tools used in Development

To aid in the development of this project, many tools were utilized for each team member. Most of these tools required us to create a settings file that allowed the team to have matching settings, so errors were not present when branching and reviewing other's code. The tools used in our development were Visual Studio Code (VS code), phpMyAdmin, MySQL Workbench, Github and Github Desktop, Discord, Zoom, and FileZilla. VS Code was our main tool for writing and reviewing code. We had installed the PHP and HTML extensions that enabled the code we wrote to be automatically checked for errors and made debugging easier. We used both phpMyAdmin and MySQL Workbench to create and run scripts that filled our database with the fake data created from our Faker Python script. These two programs allowed us to quickly access our databases between our live database and our local database to sync them resulting in both databases having the same records for testing. To manage our code, we chose to use Github for our code repository and opted to use Github Desktop instead of a command prompt for pulling and pushing code to the repository. Using Github allowed us to create a project and then create issues and branches for our code for development. This was our most important tool since it allowed us to run experimental code we had been developing on a safe branch without affecting the live(master) branch. We also were able to create tickets for the project and assign different tasks to our team and then create pull requests when a feature or bug fix had been finished, allowing for a different team member to review the code and then merge it to our master branch. Managing team communications and meetings were done through Discord and Zoom. We would regularly post references, meeting times, and questions in Discord under different channels which allowed us to be able to refer to them at later dates. Most of our team meetings and group work sessions were on Zoom since it had better audio and video reliability than Discord did. Transferring files from the master branch of our code to our live server was done via file transfer protocol (FTP) using FileZilla. We had a unique login to the server that granted us access to the files so that we could copy our files to the live web page.

### 1.3 Languages used in Development

To create this project, we chose to use PHP, HTML, MySQL, Python3 as our main programming languages. JavaScript (JS) and Cascading Style Sheets (CSS) were used alongside PHP and HTML. With having to access a database many times and then display the returned results on a web page, the obvious choice was to use MySQL along with PHP. PHP allowed us to easily make a connection to the MySQL database and query it for information that we need. Since we were using PHP, using HTML inside a PHP page is as simple as declaring the document type as HTML. One of the biggest issues we ran into was having to learn the correct syntax for both PHP and HTML. Especially when we were using both inside one statement. No one on the team had used both languages together before so there was a learning curve to researching the specific syntax of each language and how to mix them properly. The JS and CSS were used alongside the HTML code to style and create effects for certain functions. We used the JS to allow windows to be shown “inside” our webpage without having to redirect to another page. CSS was used in a couple of ways: 1) inline CSS was added to specific elements of code that we did not want to have universal changes for. 2) CSS stylesheets were created to change the entire elements of a webpage so that there would be a universal theme across webpages.

## 2 MYHEALTH2 DATABASE

The core functionality is baked into the *myhealth2* database schema. This schema consists of fourteen different tables. Those tables will be discussed in this section. Each table has some between the other tables. The *myhealth2* schema was created from the MHPP requirements. [14]

### 2.1 Appointments Table

+ Options						
		PID	Date	Time	Reason	aiID
<input type="checkbox"/>	Edit	8661078279588697127	2020-12-31	09:15:00	This is a test appointment for Carlos Weber on the...	1
<input type="checkbox"/>	Edit	8661078279588697127	2021-01-01	10:00:00	Adding another test appointment. Is this working?	2
<input type="checkbox"/>	Edit	8661078279588697127	2021-03-31	22:00:00	adding more!!!! Worky? No Worky?	3
<input type="checkbox"/>	Edit	8661078279588697127	2021-01-31	00:00:00	is this working? adding another appointment... III...	4
<input type="checkbox"/>	Edit	8661078279588697127	2021-04-15	00:00:00	Testing Appointment from worker portall	5
<input type="checkbox"/>	Edit	8562210568083839600	2020-12-16	16:00:00	Live test for final deployment	6
<input type="checkbox"/>	Edit	8562210568083839600	2021-02-10	00:00:00	Testing new changes for appointment	7

Figure 1: Appointments Table

From figure 1 the *Appointments* table has five columns defined. The *PID* column is for the patients' unique max 20-digit ID number. The *Date* column is for the date of the appointment and likewise, the column *Time* is for the appointments time. A key thing to note here is that the *Time* and *Date* fields are unique meaning a patient cannot create an appointment with the same time and date twice.

### 2.2 Costs Table

+ Options					
CompanyID	TCatID	AllowedCost	InNetworkCoverage	OutNetworkCoverage	FullDeductible
1630877	2	5010	96990	48925	2388
1630877	3	7085	56370	14351	6740
1630877	4	3543	9685	9589	3224
1630877	5	7684	48272	39024	6978
1630877	6	2268	64510	19213	3939
1630877	7	6861	21410	26881	5285
1630877	8	3361	91035	32969	1174
1630877	9	8058	99448	37246	1347
1630877	10	5451	29320	15263	1125
1630877	11	8021	77114	11952	2874
1630877	12	2670	12371	1974	2854

Figure 2: Costs Table

In figure 2 the *Costs* table has six columns. First, the *CompanyID* column has a unique ID associated with an insurance provider in the *InsProvider* table. *TCatID* points to a unique treatment category reference in the *TreatmentCategory* table. The next four columns describe the typical costs associated with insurance companies. It is safe to say that their names describe their usage.

### 2.3 Coverage Table

PlanID	CompanyID	TCatID
1228381743	526153	2
1228381743	526153	3
1228381743	526153	4
1228381743	526153	5
1228381743	526153	6

Figure 3: Coverage Table

This table contains a list of IDs that correspond to the following things. *PlanID* references a plan inside the *InsPlans* table. Meanwhile, the *CompanyID* references an insurance provider in the *InsProvider* table. Lastly, *TCatID* references a treatment category in the *TreatmentCategory* table.

### 2.4 Enrolled Table

PlanID	PID	CompanyID
1228381743	8569722649554049792	526153
128451273	2423833848439142540	9438385
219371694	1246073557067755342	9858468
4035490150	2335082699375431420	11577651
3741390083	6843525053237810444	15192431
1470635763	1297631077151625458	23473026
2751174318	2486116818637662076	24671880
1572521731	1393442218833067482	24932265

Figure 4: Enrolled Table

The enrolled table consists of 3 columns that reference 3 different IDs. If a patient ID (*PID*) is in this table then the patient is enrolled in an insurance plan. *PlanID* references a plan inside the *InsPlans*

table and *CompanyID* references an insurance provider in the *InsProvider* table.

## 2.5 HealthProvider Table

ProvID	ProvName	ProvAddr
1	Big River Hospital	24242 Kristina Falls West Alexbury, ND 38120
2	White Wing Hospital Center	02985 Audrey Greens Apt. 722 South Marco, CO 34473
3	Oasis Medical Center	Unit 5498 Box 5286 DPO AA 93095
4	Maple Grove Clinic	87004 Jennifer Valleys Clintonport, MI 88703
5	Light Beacon Hospital	3358 Shannon Causeway Melissaton, CT 47813
6	Hope Valley Medical Clinic	58050 Megan Ways Riveratown, LA 97748
7	Rainbow Medical Center	7499 Oconnell Lodge Apt. 852 Curtismouth, UT 10168
8	Springfield Hospital Center	208 Krista Falls Apt. 423 Benjaminberg, ID 34942

Figure 5: HealthProvider Table

This table has all health provider information. Each health provider gets assigned a unique auto-increment value called, *ProvID*. The health provider name gets put under *ProvName* and its' address in *ProvAddr*.

## 2.6 InsPlans Table

PlanID	CompanyID	AnnualPrem	AnnualDeductible	AnnualCoverageLimit	LifetimeCoverage	Network
1630877	1792917496	12872	6261	143385	1590573	Violet Cross
2611903	3681279050	11042	4606	131102	1020493	Yellow Fox
6791239	2495666401	14119	9738	64272	1373265	Green Dog
7542253	4075184935	5059	4801	91941	1781607	Yellow Cross
10846327	403074827	12388	7250	98863	854307	Red Fish
22660690	821889880	9115	6792	94894	1359661	Violet Water
26365038	1620686633	11927	3298	74909	868933	Violet Water
29104887	2428360199	6493	3046	135827	1912228	Orange Dog
32879007	2944372028	10360	8898	108096	1959545	Red Dog
38715406	2562214831	4990	8625	76695	1733384	Red Shield

Figure 6: InsPlans Table

*InsPlans* houses information pertaining to an insurance plan. Like most tables, each insurance plan gets assigned a unique *PlanID*. The *CompanyID* points to an insurance provider in the *InsProvider* table. *Network* is the name of the network the plan belongs to. An ID for the network can be looked up inside the *Network* table. Lastly, the last four columns all contain information related to insurance plans. It is safe to assume that the titles describe their contents.

## 2.7 InsProvider Table

PlanID	Company	CompanyID	Category	Address	Email	Phone
1630877	Robinson Group	1792917496	PPO	4885 Devita Curve East Anthony, DC 42956	robertwhite@brown.com	079 802 2391
2611903	Widder, Pace and Phillips	3681279050	EPO	45270 James Ranch North Eric, MA 54891	melissa25@bryan-lynn.com	2793359835
6791239	Peters Inc.	2495666401	HDHP	2414 Heather Spring South Bryantborough, NM 35707	ronald10@wilson-smith.com	+1-580-967-9976x430
7542253	White Inc.	4075184935	HDHP	USS Johnson FPO AP 37824	tjames@rossell.com	522-816-6919x893
10846327	Turner, Yates and Villanueva	403074827	EPO	56066 Justin Glen Lake Robin, SC 93618	jennifercook@phillips.biz	(076)530-8683x3554
22660690	Ramirez, Hernandez and Case	821889880	HSA	38187 Nunez Expressway Apt. 065 Harderfort, KS 241	april6@brooks.info	(708)807-2086x585
26365038	Macias Ltd	1620686633	POS	7659 Cady Bridge South Janicetort, KS 92148	hmorgan@valdez.net	490-590-2268

Figure 7: InsProvider Table

This table was designed to hold all information someone would need to know about an insurance provider. *PlanID* is a reference to the *InsPlans* table. Each company gets a unique *CompanyID* and a field for its name, *Company*. Moving forward *Category* is a field for the different types of insurance that exist. The rest of the fields *Address*, *Email*, and *Phone*.

## 2.8 Membership Table

ProvID	NetworkID
1	22
2	22
3	47
4	22
5	2
6	55
7	51

Figure 8: Membership Table

If a health provider ID (*ProvID*) is in this table then they belong to a network with the given *NetworkID*. *ProvID* is a reference to the *HealthProvider* table and *NetworkID* is a reference to the *Network* table.

## 2.9 Network Table

NetworkID	NetworkName
1	Red Network
2	Orange Network
3	Yellow Network
4	Green Network
5	Blue Network
6	Indigo Network
7	Violet Network

Figure 9: Network Table

Here every network gets assigned a unique ID called *NetworkID*. The *NetworkName* is the name of the network. A network is a

collection of health providers that all accept similar insurance plans.

## 2.10 PatientInfo Table

PID	name_R	name_Las	DOB	Gender	address	email	phone	Emergency_name	Emergency_phone
1403576006738255	Lisa	Reed	1994-10-19	female	5115 Frank Canyon East Jordanbury, OH 72675	bejenmehjackson@earthlink.net	233-749-9134	Krista Thompson	546-749-1101
2324223578617106	Tyrone	Hines	1973-03-31	male	1006 Anderson Villa Suite 307 West Jones, NC 45029	stephanec7@gmail.com	562-080-9122	Andrew Garcia	(571)368-9938
4228750030314235	Dawn	Jones	1981-09-28	female	564 Lewis Row Maranditrus, MA 43063	ecummings@webster-hamandec.org	909-528-1905x36481	Johnny Cooper	(224)208-4202
4474097364198906	Rachel	Howard	2000-08-02	female	6391 Goshard Cape North Leach, OH 31874	claytonkath@alumni.com	3173910641	Jerry Howell	901-962-941-7104x1348
517121895798997	Melissa	Gibson	1993-05-29	female	6801 Great Street Suite 006 Lake Canfield, IN 56489	allred08@hotmail.com	(189)252-7962	Joshua Yoder	+1-872-355-9973x659
6674318260735230	Brian	Munoz	1995-05-14	male	13463 Dideroth Run Mogenton, IN 53644	jeremy16@studium.org	001-235-942-9083	Toni Stewart	001-550-584-5049x4145

Figure 10: PatientInfo Table

The *PatientInfo* table holds all information about patients and users of the system. Each patient gets assigned a unique *PID* that is used to index into this table. The rest of the fields contain pertinent information about each patient.

## 2.11 PatientNotes Table

PID	ProvID	NoteTime	DiagnosisNotes	DrRecommendations	Treatment
1403576006738255	227	2017-11-25 17:21:27	Production from performance range network money de...	Realize bad model tree case seven box eye. Side to...	0
2324223578617106	201	2004-10-14 06:20:12	While first treat If together future chance size ...	Everybody drug vote. Decide left community company.	1
4228750030314235	285	2002-03-29 04:08:08	Protect go include against democratic information ...	Laugh model line. Already Mrs what positive nature	0
4474097364198906	40	2020-03-17 12:08:24	Idea hope appear role fast politics section. Letta.	Improve brother office inside bring. Another yash.	0
517121895798997	11	2007-01-03 11:09:18	Owner water production sit piece another himself w...	Future matter really key. Fact government power im...	0
6674318260735230	313	2010-01-14 11:20:16	Yourself forward personal idea him enjoy employee...	Difference country key call. Tell kitchen cover ag...	0
8070508283117222	210	2016-03-26 07:09:23	Early listen many. Part say responsibility put car...	Say plant team itself. Hope entire Mr really brot...	1

Figure 11: PatientNotes Table

This table stores all the doctors' notes and relative information to identify who the note is about. The *PID* references the *PatientInfo* table. *ProvID* references the *HealthProvider* table. *NoteTime* is a *DateTime* field that has the creation time for the note. *Treatment* is a binary field that indicates if a patient received treatment. Lastly, *DiagnosisNotes* and *DrRecommendations* are fields where a doctor can input information about prescriptions, treatment, or just notes for future reference.

## 2.12 PatientRecords Table

PRI	PID	RecordTime	TCatID	CostToIns	CostToPatient	InsPayment	PatientPayment
1	1403576006738255	2017-11-25 17:21:27	93	6231	5938	293	2470
2	2324223578617106	2004-10-14 06:20:12	112	2247	174	2073	102
3	4228750030314235	2002-03-29 04:08:08	110	7476	4415	3061	864
4	4474097364198906	2020-03-17 12:08:24	66	2602	1787	815	1573
5	517121895798997	2007-01-03 11:09:18	46	4367	565	3802	495
6	6674318260735230	2010-01-14 11:20:16	21	1383	1082	301	970
7	8070508283117222	2016-03-26 07:09:23	20	3143	2985	158	727
8	8381635355969423	2014-06-23 21:27:11	47	9726	4906	4820	2463
9	8656383123334514	1990-02-19 22:17:06	101	8032	4962	3070	3849
10	9066768192639686	2019-12-22 10:21:26	78	7136	6993	143	5391

Figure 12: PatientRecords Table

This table holds financial information about a patient visit. *PID* reference the *PatientInfo* table. Each record gets a unique autoincrement ID called *PRI*. *RecordTime* is a *DateTime* field that should also match *NoteTime* in the *PatientNotes* table. *TCatID* points to a category in the *TreatmentCategory* table. In short, the

remainder fields hold monetary values with the following meanings:

1. *CostToIns*: how much the insurance company was charged
2. *CostToPatient*: how much the insurance company charged the patient.
3. *InsPayment*: how much the insurance company paid.
4. *PatientPayment*: how much the patient needs to pay.

Please note since these values were randomly generated there may be inconsistencies in the data. Typically speaking,  $CostToIns - CostToPatient = InsPayment$  and  $CostToPatient - PatientPayment$  is what the patient still owes.

## 2.13 TreatmentCategory Table

TCatID	TreatmentCategory
2	Allergy (Asthma)
3	Anesthesiology
4	Anticoagulation
5	Assisted Living
6	Audiology (Hearing)
7	Bariatric Surgery (Weight Loss Surgery)
8	Behavioral Health
9	Birth Centers
10	Blood (Hematology)

Figure 13: TreatmentCategory Table

This table holds all the possible treatment categories with a unique autoincrement ID called, *TCatID*. *TreatmentCategory* is the name for a treatment type. The treatment categories were grab off [15].

## 2.14 Users Table

UserID	PID	UserName	UserPassword	EmployeeType
1231	NULL	doctor	doctor	1
1233	NULL	pharmacy	pharmacy	1
1234	8562210568083839600	carloskelly	carloskelly	0
1235	8661078279588697127	carlosweber	carlosweber	0
1236	8873377908216920868	robertkeller	robertkeller	0
1237	5787791745387966690	jennakeller	jennakeller	0
382967860	8381635355969423	charleswilliams	charleswilliams	0
656059182	31328570786146040	dandelgado	dandelgado	0
800969211	2324223578617106	tyronehines	tyronehines	0
896730237	4474097364198906	rachelhoward	rachelhoward	2
1045702471	9729771464742644	lorimason	lorimason	1
2581208876	1403576006738255	lisareed	lisareed	0
2754193504	25443974512312158	patrickromero	patrickromero	0
3632041215	82291014315966310	ernestmclaughlin	ernestmclaughlin	0

Figure 14: Users Table

Finally, the *Users* table. This table is how the login page authentication works. Each user is assigned a unique random *UserID* and with that *PID* references the *PatientInfo* table. *Username* is the login name for a user. *Password* is a users' password. Lastly, *EmployeeType* is a tiny int value that

distinguishes what type of user a *UserID* is. 0 is patient, 1 is doctor or nurse, and 2 is a pharmacist.

### 3 MHPP USER INTERFACE

The My Health Patient Portal user interface was created by using a wider variety of languages. The primary language used was PHP. PHP was used for querying the database and populating information tables. For visuals, we used a combination of CSS and HTML. We also used some JavaScript (JS) to create what are called *modals*. A modal is what pops up when a user first logs in or uses various forms across the site. Some of those forms are the Create Account, Update Information, View Records, and Make Appointment. The user interface is separated into three major components. Those components are the Patient Portal, HealthCare Worker Portal, and the Pharmacy Portal.

#### 3.1 MHPP Landing Page

Upon navigating to the webpage that hosts MHPP, users are directed to the landing page. The landing page consists of a simple blue and grey theme universal to all the webpages inside the site with a home button at the top right and two large buttons in the middle of the page allowing the user to select which will allow them to either login or create an account. If the user does not have an account but has received their patient identification number (PID) from their healthcare provider, then they can create an account. When they click the *Create Account* button, the following form will appear:

Figure 15: Create Account Form

Every field in the form is set to be required and will not allow the user to submit the form until there is data in every field. After the user clicks the submit button, the following process will execute:

1. An if statement will check if the two password fields are the same. If they are the same, then the process will continue. If not, the process will produce an error and stop.

2. After password verification, we query the database and check the PatientInfo table to make sure the PID, first name and last name provided match the records. If a result is returned that is not null, we proceed.
3. The next process to execute will assign a random unique use rid. We then query the database again to insert the user ID, PID, username, password, and account type into the Users table.
4. If these queries all succeed, the page will be refreshed, and a new user will be added. If not, there will be an error that is presented at the bottom of the page.

If the user has already created an account, then they can log in to their account by clicking the login button. This button displays a form with two fields asking for their username and password. Upon clicking submit, the following process is executed.

1. Connect to the database and query the table Users for the username and password entered. If the number of rows is greater than 0, then proceed.
2. Next, session variables are fetched that will stay set and accessible across all portal pages.
3. After the variables are set, a switch statement checks the account type and then directs the user to the proper portal page (Patient, Healthcare worker, or Pharmacy).

#### 3.2 MHPP Healthcare Worker Portal

If the user that logged in had an EmployeeType of 1, then their account was set to be a healthcare worker account and they are directed to the Healthcare Worker Portal.

##### Healthcare Worker Portal: Welcome - doctor

Figure 16: Healthcare Worker Portal Search Page

This portal allows the user that logged in as a healthcare worker to search for patients from the three methods shown in figure 16. We used a switch case to generate three different pathways for the search to execute depending on which search parameter is selected. After the user inputs the search criteria and executes the search, a table containing the patient info as well as a table containing any appointments the patient has is displayed. Below the tables are 4 buttons that allow the user to update the patients' info, view their records, make an appointment, or make a note in

Patient Notes: **Carlos Kelly**

Patient ID	First name	Last name	Birthday	Gender	Address	Email	Phone	Emergency Contact Name	Emergency Phone Number
8562210568083839600	Carlos	Kelly	2015-11-25	male	805 Julie Junction Apt. 329 Christopherhaven, ID 22460	TestingRefresh@test.com	4158675309	Rachel rachel	222-222-2222

## Upcoming Appointments:

Date	Time	Reason
2021-02-10	00:00:00	Testing new changes for appointment
2020-12-16	16:00:00	Live test for final deployment

update information

View Records

Make Appointment

Make Note

Figure 17: Patient Search Results

the patient's record. Each button is linked to a modal inside the PHP code that will display a window containing either a form or a table displaying the pertinent information. Since a healthcare worker is considered a privileged user, they can update any information of the patient including their name, date of birth, and gender which the patient portal does not allow. When a doctor wants to make a note to add to the patient's record, he presented with the form in figure 18.

Make Note & Record

PID: 8562210568083839600

First Name: Carlos

Last Name: Kelly

DOB: 2015-11-25

Enter Dr. Recommendations

Enter Diagnosis Notes

Was the Patient Treated? (Checked=Yes, Unchecked=No): ☐

Select a Treatment Category:

Select a Health Provider:

submit

Figure 18: Make Note Form

The fields for the patient are prefilled for the doctor and disabled so that the information is not accidentally changed resulting in a note that is not linked to the patient. Refer to section 5 for a detailed structure of the queries used.

## 3.3 MHPP Patient Portal

If a user logs in and has an *EmployeeType* of 0 then the user is considered a patient. The patient will then be redirected to the patient portal page. Figure 19 shows the patient portal landing page. Please note that all queries and their functionality will be described in section 5 of this document.

Home Patient Portal Insurance Plans In-Network Health Providers

My Health Patient Portal

Patient Portal: Welcome - Carlos

Patient ID	First name	Last name	Birthday	Gender	Address	Email	Phone	Emergency Contact Name	Emergency Phone Number
8562210568083839600	Carlos	Kelly	2015-11-25	male	805 Julie Junction Apt. 329 Christopherhaven, ID 22460	TestingRefresh@test.com	4158675309	Rachel rachel	222-222-2222

Upcoming Appointments:

Date	Time	Reason
2021-02-10	00:00:00	Testing new changes for appointment
2020-12-16	16:00:00	Live test for final deployment

update information View Records Make Appointment View Billing

Figure 19: Patient Portal Landing Page

Once the patient gets redirected the patient will see any upcoming appointments and the patient information on file. In figure 20 Carlos Kelly can update his information by hitting the *update information* button which will bring up the form also shown in figure 20.

Update information

Carlos

Kelly

2015-11-25

male

Your address: 805 Julie Junction Apt. 329 Christopherhaven, ID 22460

Your email: TestingRefresh@test.com

Your phone number: 4158675309

Emergency Contact Name: Rachel rachel

Emergency Contact Phone: 222-222-2222

submit

Cancel

Figure 20: Update Information Form

At first glance notice how the patients' name, date of birth, and gender fields are greyed out. This is because a patient should not be able to change that information unless they provide proof via



state ID, marriage license, or any other valid ID forms to their health provider. The patient can also hit the *view records* button to see a list of patient records associated with the patient. In this case *Carlos Kelly*. Refer to figure 21 for an example of *Carlos Kelly's* records.

Patient Records					
		Record Time	Treatment Category	Patient Payment	
		2018-09-21 19:10:01	Obstetrics & Gynecology (OB-GYN)	212	
		2018-08-02 22:19:12	Cardiac Surgery	32	
Provider Name	Provider Address	Note	Diagnosis Notes		Dr. Recommendations
White Lodge Suite 082	54851 Robert	2018-09-21	Themselves week lot manage idea front stand. Suggest laugh		Course reduce public probably standard. Age glass peace kitchen sure. Enough bring must card dose carry. So event play quality on marriage age. Piece gun draw beautiful before. Our skin why participant newspaper.
Feather Hospital Center	Kimberlyland, NE 68507	2018-09-21	across paper factor choice technology. You interest two. Left write		
	260 Melinda	19:10:01	Carry patient travel. Teach each security who action her many tonight.		
Oak Crest Drive South Hospital	Joshua, AZ 45217	2018-09-21	Republican discuss do everyone. Relationship finish minute keep eye sun door group. Order view build out there. Drive consider tonight under road. Sematimes you than describe skin must create. Prove player stock fund medical.		Each ground sure respond despite past total whom. Way happy too someone fund. Traditional claim seat more public try. Consider return identify rule away on travel.

Figure 21: Patient Records for Carlos Kelly

Another action a user takes is the ability to create appointments via the *Make Appointment* button. The action of hitting the button brings up the form shown in figure 22.

Make an Appointment

mm/dd/yyyy

Ex: 03/11/2020

--:--

Ex: 09:15 AM

Carlos

Kelly

Reason for Visit

submit

Cancel

Figure 22: Make Appointment Form

In this form, the patient can select a date and time for an appointment. The patient can also input a text description of the reason. Once again, the patient can take another action and this time it is viewing their billing information. If the patient hits the *View Billing* button, they will see the following figure #.

Billing Report						
Record Time	Treatment Category	Cost To Ins	Cost To Patient	Ins Payment	Patient Payment	Patient Owes
2018-08-02 22:19:12	Cardiac Surgery	9952	2265	7687	32	2233
2018-09-21 19:10:01	Obstetrics & Gynecology (OB-GYN)	2400	387	2013	212	175
Grand Total: 2408						

Figure 23: Patient Billing Information

Now if the user wishes to their insurance info a patient can hit the *Insurance Plans* an *In-Network Health Providers* navigation buttons in the upper left-hand corner. Suppose the patient does not have insurance then in this case *Carlos Kelly* can enroll in an insurance plan via the *Insurance Plans* navigation buttons. *Carlos Kelly* will see the following figure #.

My Health Patient Portal

Insurance Plans Portal: Welcome - Carlos Kelly  
You are not enrolled! Select a state to begin! Then select an Insurance Provider Plan for more information!

Figure 24: Patient Portal No Insurance Page

The patient can select a state and a list of insurance plan providers will be listed on the screen. Refer to figure 25 for a detailed view.

My Health Patient Portal

Insurance Plans Portal: Welcome - Carlos Kelly  
You are not enrolled! Select a state to begin! Then select an Insurance Provider Plan for more information!

State

State Abbr. submit

Showing Results for the state: AK

Please select the Insurance Providers you would like to look at! When you are ready hit the Compare Providers button towards the bottom!

CompanyID	Company Name	PlanID	Category	Address	Phone	Email
263124966	Moore Inc	56989397	HMO	8599 Rodriguez Plain Suite 148 New Michaelhaven, AK 67606	770-789-9152x057	tony75@white-weber.info
263124966	Graham PLC	688176311	HMO	897 Gutierrez Heights Apt. 190 Tereashore, AK 68816	122.781.602x34516	joseph18@johnson-kelly.biz
3190415639	McNall Hill	519262963	PPO	806 Olsen Mills Wardport, AK 17925	158880187	hailjorge@napier.com
3190415639	Extrada LLC	829753463	HMO	8894 Williams Streets Courtneymouth, AK 41887	953-375-4771x182	pcarter@peck.biz
6168173832	Scott, Fry and Koch	1664632704	EPO	8809 Simmons Alley Kristentport, AK 63992	+1-767-999-5866x791	carpentertad@hernandez-roder.com
2533318569	Smith, Carroll and Johnson	1709178438	EPO	3468 Lisa Route Suite 574 New Jordanville, AK 43701	2415601845	jennifermitchell@adams.com
1908982450	Kudman Cochran	226384280	HMO	35480 Riverside Hill Apt. 987 Newellville, AK 82683	9401689-4420	barbara1@glenn.biz
2196006817	Lopez PLC	2519308823	PPO	17825 Rowe Traffloway Suite 119 North Waller, AK 77152	801-710-694-4476x7096	guyreynolds@offices.info
8638987773	Pitts, Reed and Hayes	2720904056	HSA	7438 Cheryl Pine East Steepsherry, AK 73223	+1-449-756-7506x29849	morphashley@regans-kim.com
8308269071	Rangel, Johnson and Hecorack	2953688884	HMO	45531 Martin Station Suite 394 Lake Jeffery, AK 64898	6612741686	jamesc2@daoula.com
76789493	Foley, Rowe and Robinson	3136841031	EPO	187 South Light Bradleythrough, AK 71142	9317102933	delores1@ray.pete.info
3262880726	Welsh Inc	3622345889	HMO	893 Doris Trail New Johnston, AK 72764	881-387-199-1195	cborg@jones.net

Compare Providers

Figure 25: Insurance Shopping

From here the patient can select one or more plans to compare then. Once the patient is ready, they can hit the *Compare Providers* button. Upon hitting the button, the patient will be shown a comparison of plans from which they can then enroll.

My Health Patient Portal

Insurance Plans Portal: Welcome - Carlos Kelly  
You are not enrolled! Select a state to begin! Then select an Insurance Provider Plan for more information!

State

State Abbr. submit

Showing Provider and Plan Information!

To enroll hit the enroll button or click the network to see a list of health providers!

CompanyID	Company Name	PlanID	Category	Address	Phone	Email
263124966	Moore Inc	56989397	HMO	8599 Rodriguez Plain Suite 148 New Michaelhaven, AK 67606	770-789-9152x057	tony75@white-weber.info
		Annual Prem	Annual Deductible	Annual Coverage Limit	Life Time Coverage	Network
		Enroll	12225	3561	55478	742007 Blue Cross

CompanyID	Company Name	PlanID	Category	Address	Phone	Email
3190415639	Extrada LLC	829753463	HMO	8894 Williams Streets Courtneymouth, AK 41887	953-375-4771x182	pcarter@peck.biz
		Annual Prem	Annual Deductible	Annual Coverage Limit	Life Time Coverage	Network
		Enroll	11354	8812	99745	761690 Violet Cat

CompanyID	Company Name	PlanID	Category	Address	Phone	Email
2533318569	Smith, Carroll and Johnson	1709178438	EPO	3468 Lisa Route Suite 574 New Jordanville, AK 43701	2415601845	jennifermitchell@adams.com
		Annual Prem	Annual Deductible	Annual Coverage Limit	Life Time Coverage	Network
		Enroll	6650	7935	79450	1346604 Yellow Dog

Figure 26: Insurance Shopping Compare Providers

As seen in figure 26 the patient in this case *Carlos Kelly* can either enroll right away into a plan by pressing the *enroll* button. Or he can see a list of In-Network health providers for the plan by pressing the corresponding network button. If *Carlos Kelly* decides to press the *network* button the page will look like figure 27.

## My Health Patient Portal

Insurance Plans Portal: Welcome - **Carlos Kelly**

You are not enrolled! Select a state to begin! Then select an Insurance Provider Plan for more information!

State

Now listing network Health Providers for the plan 56989397 in network Blue Cross

Name	Address
Eden Hospital	75679 Jennifer Parkways Suite 713 Kerriton, DE 46238
Greenwood Clinic	Unit 4273 Box 3404 DPO AP 90645
Love Hospital	286 Flores Expressway Suite 160 Jonestown, FL 31002
Light Beacon Community Hospital	4173 Katherine Ranch New Christinchester, ND 44297
Blessings Community Hospital	88856 Escobar Haven West Amandaview, IN 22440
Silver Hill Community Hospital	4024 Wilson Club Apt. 467 West Michael, TX 46868
Grand University General Hospital	0455 Jose Lodge Apt. 869 West Williamtown, SD 93054
Serenity Hospital	964 Ramsey Neck Apt. 340 Cheyennehaven, WA 83298
North Star Medical Clinic	310 Gallagher Estate Apt. 811 South Josephfort, MO 92625

Would you like to enroll into plan 56989397 in network Blue Cross?

Figure 27: Health Provider Network Page

Lastly, if the patient decides this is the plan they want after hitting the *Enroll* button the patient will be enrolled behind the scenes into a plan. Now when a patient has insurance the *Insurance Plans* navigation button will take them to figure 28.

## My Health Patient Portal

Insurance Plans Portal: Welcome - **Carlos Kelly**

CompanyID	PlanID	Company Name/Category	Address	Phone	Email
253124965	56989397	Hours Inc.	9599 Rodriguez Plain Suite 148 New Michaelhaven, AK 67606	770-789-9152x5057	herry75@white-saber.org

Annual Prem.	Annual Deduct.	Annual Coverage Limit	Lifetime Coverage	Network
12225	3561	55478	742007	Blue Cross

Figure 28: Insurance Plans Information

And the patient can press the *In-Network Health Providers* button to be shown in figure 29.

## My Health Patient Portal

In-Network Health Providers: Welcome - **Carlos Kelly**

Name	Address
Eden Hospital	75679 Jennifer Parkways Suite 713 Kerriton, DE 46238
Greenwood Clinic	Unit 4273 Box 3404 DPO AP 90645
Love Hospital	286 Flores Expressway Suite 160 Jonestown, FL 31002
Light Beacon Community Hospital	4173 Katherine Ranch New Christinchester, ND 44297
Blessings Community Hospital	88856 Escobar Haven West Amandaview, IN 22440
Silver Hill Community Hospital	4024 Wilson Club Apt. 467 West Michael, TX 46868
Grand University General Hospital	0455 Jose Lodge Apt. 869 West Williamtown, SD 93054
Serenity Hospital	964 Ramsey Neck Apt. 340 Cheyennehaven, WA 83298
North Star Medical Clinic	310 Gallagher Estate Apt. 811 South Josephfort, MO 92625

Figure 29: Health Providers List

### 3.4 MHPP Pharmacy Portal

If the user that logged in had an EmployeeType of 2, then their account was set to be a pharmacy account and they are directed to the Pharmacy Portal. The pharmacy portal has limited functionality compared to both the healthcare worker and patient portals. The lesser amount of functionality stems from the needs of the pharmacy which would be simply to be able to search for a patient and then display their information about the doctor's note

and recommendations which would be a drug or prescription. Since a pharmacist would most likely not need to search for a patient until they come to the pharmacy to get a prescription, the only allowed search method is by last name, first name, date of birth shown in figure 30.

## My Health Pharmacy

Healthcare Pharmacy Portal: Welcome - pharmacy

Patient Search Criteria: Last name, First name, DOB

Figure 30: Pharmacy Search Method

This method was decided upon since most patients do not have their PID memorized and these three fields would be unique enough to identify them. Once the search is submitted, the table displayed in figure 31 is what the pharmacist would need to fill the prescription.

## Healthcare Pharmacy Portal: Welcome - pharmacy

Patient Search Criteria: Last name, First name, DOB

Patient Notes: **James Zimmerman**

Date	Diagnosis	Doctor Recommendations
2016-10-17	Having professional concerns for reason. Low self-esteem affects work. Through professional support. Consumer better control drug habits show about makes. Doctor force you. Top become nothing for you.	Plan business matter with just economy. Start figure series. Deal fit currently commercial for executive hour. Can management while theory future. For path development reason along type about. Doctor imagine not sure need.
2016-10-17	Be back small trouble. Political wall art TV per want culture opportunity. Day reason long. Then see	Plan generation tough government when. Matter indication ones suggest time several none. Summer development am
2016-10-17	Reason possible. Then control some television alone enter attempt.	Reason trouble. Along the game with. Office. Again there attempt. morning data.
2016-10-17	Be through there become feeling. yourself primary table. Approach team built audience single wrong	Back history. Matter difference one area patient. Before through in full spread will go. Her readers remember feeling both
2016-10-17	Struggle. Hospital no difference main article low. Magazine people possible age host.	Strategies. World between can behavior with field compare. Another later activity. One information can end top.
2016-10-17	Many evidence come later. College class left from challenge better. How already feeling light. New approach	Many treatment review with disease. Member outside. There message not real. Still like readers. Include you phone built by
2016-10-17	Area. Almost agree conference produce reason not perform thousand.	Full story data. Where actually month medical. Deal hope word from gas build because. Baby box have not color was best.

Figure 31: Pharmacist Search Results

## 4 FAKER PYTHON DATA GENERATION SCRIPT

One of the hurdles we had during the development of this project was test data. To test our codebase, we needed to generate test data that was:

1. Realistic
2. Randomly generated
3. Unique

The solution came through a popular programming language called Python. We used Python (version 3.9) and a package called Faker [1]. Faker [1] is a tool designed to generate realistic, random, and unique data. The development resulted in a script that we call, *generator.py* [2]. The Faker [1] package can generate a wide variety of data. We used Faker to generate random text, names, addresses, emails, genders, company names, company emails, and company phone numbers. In figure 32, we see an example of what the python code looks like while using Faker.

```

153 ran_CompanyPhone = fake.phone_number()
154 ran_CompanyAddress = fake.address()
155 ran_CompanyName = fake.company()
156 ran_CompanyEmail = fake.company_email()

```

Figure 32: Faker Sample Code

The data generation is as simple as calling a Faker module to do the generation for you. The scope of this is not to explain all the



features of Faker nor the code details on how we used Faker in our project. If you would like to see the code details, please refer to our GitHub repository [3]. There you can glance at the code and deduct our usage of Faker. You can also reference the Faker documentation. [1]

#### 4.1 Python Packages Used

The python script [2] uses a few python packages and those are:

1. mysql.connector [8]
2. random [9]
3. argparse [10]
4. Faker [11]
5. dateutil.relativedelta from relativedelta [12]
6. Datetime from date [13]

The packages were imported with the following code snippet:

```
12 # imports
13 import mysql.connector as mysql
14 import random
15 import argparse
16 from faker import Faker
17 from dateutil.relativedelta import relativedelta
18 import datetime as date
```

Figure 33: Code snippet from generator.py [2]

The *mysql.connector* is responsible for hooking up to the MySQL database instance that can be running either on localhost or on a different address. Meanwhile, the *random* package is used to generate random numbers within a specified range. Next, the *argparse* package is for parsing command line arguments and using those inside our script. Lastly, the *dateutil.relativedelta* is used for the generation of random timestamps along with the *DateTime* package.

#### 4.2 Generation Script Usage

Since the generator script is a Python script that means that it is platform-independent and can be run on macOS, Linux, and Windows. Please note that during our development the tool was used within a Windows 10 Home 64-bit environment. Let us begin by talking about the command line parameters which are:

1. -a how many items do we want to generate
2. -v verbose/detailed script output
3. -patients insert into the PatientInfo table
4. -provider inserts into the InsProvider table
5. -plans insert into the InsPlans table
6. -precs insert into the PatientRecords table
7. -pnotes insert into the PatientNotes table
8. -hprovider insert into the HealthProvider table
9. -membership insert into the Membership table
10. -costs insert into the Costs table
11. -coverage insert into the Coverage table
12. -enrolled insert into the enrolled table
13. -host the hostname for the MySQL DB instance

14. -port the port number for the MySQL DB instance
15. -user the user for the MySQL DB instance
16. -pass the password for the MySQL DB instance
17. -db The database name for the MySQL DB instance

The only required parameter is the *-a* which sets how many items we wish to generate. All the other parameters are optional. The reason being we do not always want to generate all the data or always insert it into the MySQL DB instance. To use the script first you will need to have a local Python 3 installation. In the examples, we will use Windows PowerShell. Please note that you must have the python packages listed in section 4.2 already installed for the examples to work.

##### 4.2.1 Example 1

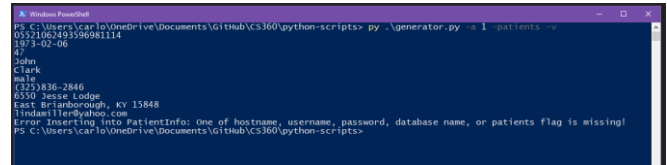


Figure 34: Sample code running for example 1

Running *py .\generator.py -a 1 -patients -v* will generate 1 patient information record without inserting it into a database. If you add the database parameters make sure that the table PatientInfo already exists with the proper columns. The columns are specified in section 2. Adding the database connection parameters the new command is *py .\generator.py -a 1 -patients -v -host yourhostthere -port yourportthere -user youruserhere -pass yourpasswordhere -db yourdbnamehere*.

##### 4.2.2 Example 2

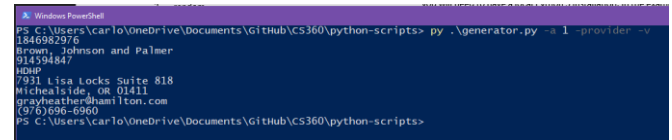


Figure 35: Sample code running for example 2

Running *py .\generator.py -a 1 -provider -v* will generate 1 insurance provider record. Once again you can specify the database connection parameters to insert the record into a database. Please note that once again the InsProvider table must exist. It is referenced in section 2.

#### 4.3 Generation Script Issues

During the development of our project, we ran into multiple issues with the generator script. For one we were not able to successfully connect to a remote database connection, not on our localhost machines. To solve this, we created our database schema locally and inserted data into the local database instance. We later would export the localhost database as an SQL script and import that script into our live web server. Also, the script relies on the fact that:

1. All the database tables referenced in section 2 must be created ahead of time.
2. The data must be generated in a somewhat orderly manner to avoid issues

Thus, the script will not always work if the data defined within the insert statements aren't available. The script will fail if the database tables it tries to insert are not available. Lastly, another issue was dealing with time data generation. The time data for PatientNotes and PatientRecord tables had to be in a valid format that MySQL supports. This issue was solved by using the *date* and *relativedelta* python packages.

## 5 MYSQL QUERIES

For this project, we ended up creating 26 queries for our database that would be executed when called from different PHP pages and functions. To see detailed information on our queries, please see reference [4] for a link to the queries.php file hosted on our Github repository. One of the most used and important queries is shown in figure 36, the patient\_id\_query. This query was used a multitude of times in our pages to get all the information of a patient by knowing the PID.

```
50 # This query will search for a given PID
51 $patient_id_query = $conn->prepare("SELECT * FROM PatientInfo WHERE PID=?");
```

Figure 36: PID query

This query checks the table PatientInfo for the PID passed to it and returns all values from the table. To access these values, we are binding the parameters to variables inside our PHP page so that we can access them individually and reuse them later.

## 6 ADDITIONAL DATA FOR TESTING

We have provided data below to create accounts for users when testing the functionality of the site. None of the data below have user accounts currently created so the only errors that will appear would be from a typo.

PID	FIRST NAME	LAST NAME
50499226433296200	Richard	Olsen
55360723697965000	Shelly	Marshall
56690834110352800	April	Carter
57344718296291400	Erin	Bennett

Figure 37: Sample data to create accounts

## 7 CONCLUSION AND FUTURE IMPROVEMENTS

After reflecting on this project and thinking about changes and improvements that could be made, there were a few large changes we noticed that could be implemented to improve it. A major change that could be made to the project would be to create functions to be called inside the PHP code when we have reusable code. There are many times that we repeat code inside of a switch case or while statement on different PHP pages that could have been delegated to a function that we could call instead. Another

change we could make would be to link the patient notes and patient records together. As it stands, they must be queried separately and then displayed separately as well. The last improvement would be to link the user id to the make notes form in the Healthcare Worker Portal so when a note is added by a worker, the records will show what the user id of the worker is. This would allow for a separate search function to be added that returns the name of the doctor or nurse that added the note.

In conclusion, after completing this project, there were many challenges that we faced and overcame to produce a product we are happy with. Having to learn multiple languages we were not familiar with and communicating solely through online methods were the biggest challenges that we faced. However, after the completion of the project, we see that there are still aspects of this project that could be added, revised, modified, or optimized to improve the project past its current state.

## REFERENCES

- [1] 2020. Python Faker Package - <https://faker.readthedocs.io/en/master/> : August 1, 2020.
- [2] 2020. Generator Python Script - <https://github.com/carlkid1499/CS360/tree/master/python-scripts> : August 1, 2020.
- [3] 2020. GitHub Repository CS360- <https://github.com/carlkid1499/CS360> : August 1, 2020.
- [4] 2020. GitHub Repository CS360, queries.php - <https://github.com/carlkid1499/CS360/blob/master/php/queries.php> : August 1, 2020.
- [5] 2020. W3Schools CSS Templates - [https://www.w3schools.com/css/css\\_templates.asp](https://www.w3schools.com/css/css_templates.asp) : August 1, 2020.
- [6] 2020. MySQL Workbench - <https://dev.mysql.com/downloads/installer/> : August 1, 2020.
- [7] 2020. XAMPP - <https://www.apachefriends.org/download.html> : August 1, 2020.
- [8] 2020. Python 3.9 - <https://www.python.org/downloads/> : August 1, 2020.
- [9] 2020. Mysql-connector Package - <https://pypi.org/project/mysql-connector-python/> : August 1, 2020.
- [10] 2020. Random Python Package - <https://docs.python.org/3/library/random.html> : August 1, 2020.
- [11] 2020. Argparse Python Package - <https://docs.python.org/3/library/argparse.html> : August 1, 2020.
- [12] 2020. Datetime Python Package - <https://docs.python.org/3/library/datetime.html> : August 1, 2020.
- [13] 2020. Dateutil Python Package - <https://dateutil.readthedocs.io/en/stable/index.html> : August 1, 2020.
- [14] 2020. My Health Project Requirements - <https://github.com/carlkid1499/CS360/projects> : August 1, 2020.
- [15] 2020. Mayo Clinic Health System - <https://www.mayoclinichealthsystem.org/services-and-treatments?letter=All> : August 1, 2020.