The Clinicians Guide: Developer Docs

A manual and tracking tool for mental health treatment

Christopher Miller
Danny Kumpf
Joshua Cullings
Rostik Mertz
Philip Neill

Made in conjunction with ESOF 423 at Montana State University

April 27, 2019

1 Introduction

Tracking the course of treatment of patients seeking aid for mental health issues is crucial to a successful course of treatment. Knowing not only where the patient has been in their past visit but also getting a macroscopic view where they started and how they have progressed throughout all of their treatment is a powerful tool for both doctor and patient. Additionally, taking the tedious work out of calculating patient progress enables clinicians to focus more on their patient while still achieving reliable notation. The Clinical Guidelines app aims to accomplish all of this for clinicians and their patient treatment workflows.

2 Database

2.1 Tables

Doctor Information

Purpose

This table is used to store all pertinent information regarding the doctor/user. This includes the current patients for the given doctor, first and last name, and the DoctorID. As a new doctor is added the user is queried for the FirstName, LastName, UserName, and password.

Fields:

Primary Key: DoctorID
The unique primary key for this table. A non-null integer that is auto-incremented when a new doctor has been added to the database.

FirstName
A non-null string of characters set aside for the first name of the doctor. *Maximum size:* 30 characters

LastName
A non-null string of characters set aside for the last name of the doctor. *Maximum size:* 30 characters

Username
The unique non-null string of characters set aside for the doctor to sign in with. *Maximum size:* 45 characters

Foreign Key: PatientID
An integer, that can be left null, to identify a doctor's patients by. Also a foreign key from PatientInformation.

Medication Information

Purpose
This table is used to store information regarding medication as well as the dosage and associated diagnosis for a given medication.

Fields:

Primary Key: MedicationID
The unique primary key for this table. A non-null integer that is auto-incremented when a new medication is added to the database.

MinimumDosage
A string of characters, that can be left null, set aside to describe the smallest dose for each drug. *Size:* MEDIUMTEXT

MaximumDosage
A string of characters, that can be left null, set aside to describe the largest dose for each drug. *Size:* MEDIUMTEXT

Diagnosis
A string of characters, that can be left null, set aside to describe the diagnosis that is associated with each drug. *Size:* MEDIUMTEXT

Patient Information

Purpose

This table is used to store all pertinent personal information for a given patient. As a new patient is added the PatientID is auto-incremented in the database and the user is queried for the FirstName and SurName. Other information, such as CurrentDose, MedicationID, and NextVisit, may be added later through scheduling or prescription of medication.

Fields:

Primary Key: PatientID

The unique primary key for this table. A non-null integer that is auto-incremented when a new patient is added to the database.

FirstName

A non-null string of characters set aside for the first name of the patient. *Maximum size:* 30 characters

Surname

A non-null string of characters set aside for the surname of the patient. *Maximum size:* 30 characters

Diagnosis

A string of characters, that can be left null, set aside to describe the diagnosis given to the patient. *Size:* MEDIUMTEXT

Foreign Key: MedicationID

An integer, that can be left null, to identify the medication the patient is on. Also a foreign key from MedicationInformation.

CurrentDose

A string of characters, that can be left null, set aside to describe the largest dose for each drug. *Maximum Size:* 15 characters

LastVisit

A date saved from the patient's last visit or left null if there is not last visit.

NextVisit

A date saved from the patient's next visit or left null if there is has not been a next visit scheduled.

Foreign Key: DoctorID

An integer, that can be left null, to identify a patients's doctor by. Also a foreign key from DoctorInformation.

Conflicting Medication

Purpose

This table is used to store all medication that conflicts with the medication in question. The ConflictingID stores the id for the medication that the doctor is looking for and the MedicationID is the medication that conflicts with it. There can be repeated entries with the same ConflictingID, however they must have a different MedicationID. This way you can query the database to get a list of all conflicting medication for a specific id.

Fields:

Primary Key: ConflictingID

The unique primary key for this table. A non-null, auto-incremented, integer that is the is used to normalize the table and allow for a medication to conflict with multiple medications.

Foreign Key: MedicationID1

An integer, that can be left null, to identify which medication conflicts with a specific medication. Also a foreign key from MedicationInformation.

Foreign Key: MedicationID2

An integer, that can be left null, to identify which medication conflicts with a specific medication. Also a foreign key from MedicationInformation.

Visit

Purpose

This table is used to store the visitation information for a given patient. This includes the DoctorID, NextVisit, and LastVisit.

Fields:

Primary Key: VisitID

The primary key for the table. It is assigned from PatientID in the PatientInformation table and serves to identify the correct visit entry for a given patient.

Foreign Key: DoctorID

A foreign key inherited from the DoctorInformation table. Included for a doctor to be able to see his/her upcoming appointments.

LastVisit

A date saved from the patient's last visit or left null if there is not last visit.

NextVisit

A date saved from the patient's next visit or left null if there is has not been a next visit scheduled.

Prescription

Purpose

This table is designed to take in a PatientID, MedicationID, CurrentDosage, and the Diagnosis it was prescribed for. It allows for doctors to prescribe multiple prescriptions to a single patient.

Fields:

Primary Key:PrescriptionID

The unique primary key for this table. A non-null integer that is auto-incremented when a new prescription is added to the database.

Foreign Key: PatientID

An integer, that can be left null, to identify the patient the prescription is made for. Also a foreign key from PatientInformation.

Foreign Key: MedicationID

An integer, that can be left null, to identify the medication that is being prescribed. Also a foreign key from PatientInformation.

Diagnosis

A string of characters, that can be left null, set aside to describe the diagnosis that is associated with each with the patient in question. *Size:* 30 characters

CurrentDosage

A string of characters, that is required, set aside to denote the amount of medication prescribed to the patient in question. *Size:* mediumtext

Diagnosis

Purpose

This table is used to store all diagnoses associated with a given patient. There can be repeated entries with the same PatientID, however they must have a different Diagnosis. This way you can query the database to get a list of all related diagnoses.

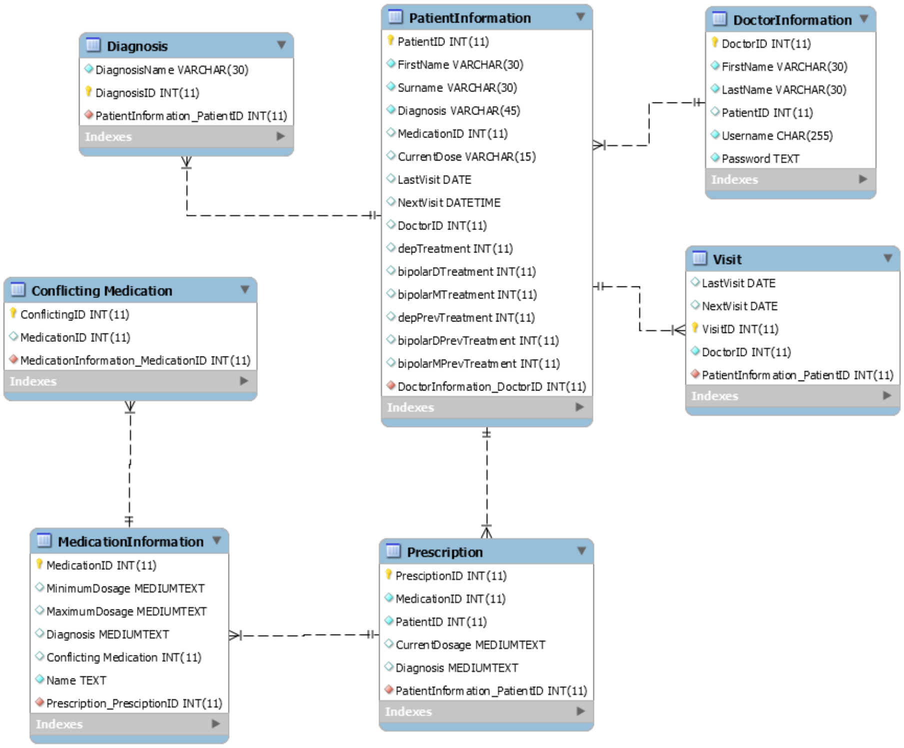
Fields:

PatientID
The primary key for this table. A non-null integer used to identify the patient that has a specific diagnosis.

Diagnosis
A string of characters, that can be left null, set aside to describe the diagnosis that is associated with each with the patient in question. *Size:* 30 characters

2.2 Entity-Relationship Diagram

Purpose
Located here is the entity relationship diagram for the application, this ERD can be used for help visualising the various views as well as well as maintaining data integrity. Should changes to the database occur it is highly recommended that the ERD is updated with haste, as the application expands this helps keep a very concise view.



Software
The software used to create this ERD was MySQL Workbench, this combined with the previous sections on fields will allow a duplicate database to be created should it be required but also supply admins with the information they need to fully manipulate the database.

3 Testing

Testing all integrated and automated using travis-ci.

3.1 phpunit

This phpunit implementation uses stubs and mocks to replicate server interactions. The stubs and mocks can be found in /tests/stubs/ and the tests can be found in /tests/testSuite/.

To run the tests, and to get the test coverage run:

```
./vendor/bin/phpunit –bootstrap vendor/autoload.php tests/
```

In the main directory. The test coverage can by found by opening clover.xml in the main directory and scrolling down to the bottom. Coverage is done with coverage-clover and is updated every time the above command is run. Coverage is at 74% for our stubs and mocks.

3.2 Selenium

Selenium is used as an additional precaution to ensure that the site is up and running correctly. The Selenium file with all tests can be found here in the repository:tests/seleniumSuite/the_clinic.side The individual tests can be found at /tests/selenium if you want to change any of the tests.

4 Webpages

4.1 Bipolar

If you navigate to the /webpages/bipolar/ directory of the repository you will find php files with html and php scripts in them. These navigate and display the bipolar diagnosis and medication decision trees.

4.2 Creating Treatment Diagnosis Algorithms

The code for the user interface for creating treatment diagnosis algorithms can be found in the /webpages/js/ directory. The files that deal with storing the information can be found in the /webpages/createAlgo/ directory. These decision trees are stored in json files on the server.

4.3 Medication

The creation of new medications, as well as dealing with using old medications, and setting conflicting medications can be found in the /webpages/medications/ directory.

4.4 User Interface

The main pages of the websites User Interface are displayed using html, and can be found in /webpages/ without going deeper into the directories. For the creating treatment diagnosis algorithms the webpage is displayed using javascript.