Clinic Management System

Technical Manual

INSTITUTE *of* TECHNOLOGY CARLOW

18th April 2018
BSc (Hons) Software Development

Name: Ryan Donoghue
Year: 4th year
Student ID: C00194829
Supervisor: Paul Barry

Installation Instructions

Development Language Environment Setup

In order to develop and run the system, the environment must be setup. As the system was developed on Windows, this document will only outline the steps required to get up and running on Windows.

1. Install Elixir v1.5.2 or greater (Location: at https://elixir-lang.org/install.html)

2. Install PostgreSQL (Location: https://www.postgresql.org/)
    a. NOTE: database username must be “postgres” and database password must be “postgres”

3. Install Node.js (Location: https://nodejs.org/en/)

4. Install Git (Location: https://git-scm.com/)

5. Install NPM (https://www.npmjs.com/)

6. Install Phoenix (Run: mix archive.install https://github.com/phoenixframework/archives/raw/master/phoenix_new-1.2.5.ez)

7. Install Brunch (Run: npm install -g brunch)

8. Run NPM (Run. npm install)

Necessary Packages

These packages are necessary for some key functionality within the app, but are included in source control, therefore do not have to be manually installed or configured.

- Comeonin
- Cloak
- Drab

Running the System

1. Download the system from source control (Run: git clone https://github.com/r-donoghue/CMS.git)
2. Move to downloaded folder (Run: cd CMS)
3. Compile dependencies (Run: mix deps.compile)
4. Get dependencies (Run: mix deps.get)
5. Create database (Run: mix ecto.create)
6. Migrate database changes (Run: mix ecto.migrate)
7. Add default values to database (Run: mix run priv/repo/seeds.exs)
8. Install brunch (Run: npm install brunch)
9. Install npm (Run: npm install)
10. Start server (Run: mix phoenix.server)

The server should now be active, and can be accessed at http://localhost:4000/. When the system was being set up, there should now be multiple records added to all tables of the database. For demo purposes there is imitation patient data, and two imitation users.

Default user details:

Username	Password
deirdre	12345678
admin	12345678

Controllers

Agecalc.ex

```
defmodule AgeCalc do
   @spec age(Ecto.Date.t, atom|{integer, integer, integer}) :: integer
   def age(%Ecto.Date{day: d, month: m, year: y}, as_of \\ :now) do
     do_age({y, m, d}, as_of)
   end
    ############
   # Internals
   ############
   @doc false
   def do_age(birthday, :now) do
     {today, _time} = :calendar.now_to_datetime(:erlang.now)
     calc_diff(birthday, today)
   end
   def do_age(birthday, date), do: calc_diff(birthday, date)
    @doc false
   def calc_diff({y1, m1, d1}, {y2, m2, d2}) when m2 > m1 or (m2 == m1 and d2 >=
d1) do
     y2 - y1
   end
   def calc_diff({y1,_,_}, {y2,_,_}), do: (y2 - y1) - 1
 end
```

Auth.ex

```elixir
defmodule Cmsv1.Auth do
    import Plug.Conn
    import Comeonin.Bcrypt, only: [checkpw: 2, dummy_checkpw: 0]

    def init(opts) do
        Keyword.fetch!(opts, :repo)
    end


    def call(conn, repo) do
        user_id = get_session(conn, :user_id)
        user = user_id && Cmsv1.Repo.get(Cmsv1.User, user_id)
        assign(conn, :current_user, user)
    end

    # login
    def login(conn, user) do
        conn
        |> assign(:current_user, user)
        |> put_session(:user_id, user.id)
        |> put_session(:clinic_id, user.clinic_id)
        |> put_session(:user_name, user.name)
        |> put_session(:level, user.level)
        |> configure_session(renew: true)
    end

    # login using username and password
    def login_by_username_and_pass(conn, username, given_pass, opts) do
        repo = Keyword.fetch!(opts, :repo)
        user = repo.get_by(Cmsv1.User, username: username)
        cond do
        user && checkpw(given_pass, user.password_hash) ->
```

```
            {:ok, login(conn, user)}
        user ->
            {:error, :unauthorized, conn}
        true ->
            dummy_checkpw()
            {:error, :not_found, conn}
        end
        end

    # logout
    def logout(conn) do
        configure_session(conn, drop: true)
    end

end
```

C_doctor_controller.ex

```elixir
defmodule Cmsv1.CDoctorController do
 use Cmsv1.Web, :controller

 alias Cmsv1.CDoctor

 def index(conn, _params) do
   cdoctors = Repo.all(CDoctor)
   render(conn, "index.html", cdoctors: cdoctors)
 end

 def new(conn, _params) do
   changeset = CDoctor.changeset(%CDoctor{})
   render(conn, "new.html", changeset: changeset)
 end

 def create(conn, %{"c_doctor" => c_doctor_params}) do
   changeset = CDoctor.changeset(%CDoctor{}, c_doctor_params)

   case Repo.insert(changeset) do
     {:ok, _c_doctor} ->
       conn
       |> put_flash(:info, "C doctor created successfully.")
       |> redirect(to: c_doctor_path(conn, :index))
     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset)
   end
 end

 def show(conn, %{"id" => id}) do
   c_doctor = Repo.get!(CDoctor, id)
   render(conn, "show.html", c_doctor: c_doctor)
 end
```

```
def edit(conn, %{"id" => id}) do
  c_doctor = Repo.get!(CDoctor, id)
  changeset = CDoctor.changeset(c_doctor)
  render(conn, "edit.html", c_doctor: c_doctor, changeset: changeset)
end

def update(conn, %{"id" => id, "c_doctor" => c_doctor_params}) do
  c_doctor = Repo.get!(CDoctor, id)
  changeset = CDoctor.changeset(c_doctor, c_doctor_params)

  case Repo.update(changeset) do
    {:ok, c_doctor} ->
      conn
      |> put_flash(:info, "C doctor updated successfully.")
      |> redirect(to: c_doctor_path(conn, :show, c_doctor))
    {:error, changeset} ->
      render(conn, "edit.html", c_doctor: c_doctor, changeset: changeset)
  end
end

def delete(conn, %{"id" => id}) do
  c_doctor = Repo.get!(CDoctor, id)

  # Here we use delete! (with a bang) because we expect
  # it to always work (and if it does not, it will raise).
  Repo.delete!(c_doctor)

  conn
  |> put_flash(:info, "C doctor deleted successfully.")
  |> redirect(to: c_doctor_path(conn, :index))
end

plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

defp authenticate(conn, _opts) do
  if conn.assigns.current_user do
  conn
```

```
    else
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Clinic_controller.ex

```
defmodule Cmsv1.ClinicController do
 use Cmsv1.Web, :controller

 alias Cmsv1.Clinic

 def index(conn, _params) do
   clinics = Repo.all(Clinic)
   render(conn, "index.html", clinics: clinics)
 end

 def new(conn, _params) do
   changeset = Clinic.changeset(%Clinic{})
   render(conn, "new.html", changeset: changeset)
 end

 def create(conn, %{"clinic" => clinic_params}) do
   changeset = Clinic.changeset(%Clinic{}, clinic_params)

   case Repo.insert(changeset) do
     {:ok, _clinic} ->
       conn
       |> put_flash(:info, "Clinic created successfully.")
       |> redirect(to: clinic_path(conn, :index))
     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset)
   end
 end

 def show(conn, %{"id" => id}) do
   clinic = Repo.get!(Clinic, id)
   render(conn, "show.html", clinic: clinic)
 end
```

```
def edit(conn, %{"id" => id}) do
  clinic = Repo.get!(Clinic, id)
  changeset = Clinic.changeset(clinic)
  render(conn, "edit.html", clinic: clinic, changeset: changeset)
end

def update(conn, %{"id" => id, "clinic" => clinic_params}) do
  clinic = Repo.get!(Clinic, id)
  changeset = Clinic.changeset(clinic, clinic_params)

  case Repo.update(changeset) do
    {:ok, clinic} ->
      conn
      |> put_flash(:info, "Clinic updated successfully.")
      |> redirect(to: clinic_path(conn, :show, clinic))
    {:error, changeset} ->
      render(conn, "edit.html", clinic: clinic, changeset: changeset)
  end
end

def delete(conn, %{"id" => id}) do
  clinic = Repo.get!(Clinic, id)

  # Here we use delete! (with a bang) because we expect
  # it to always work (and if it does not, it will raise).
  Repo.delete!(clinic)

  conn
  |> put_flash(:info, "Clinic deleted successfully.")
  |> redirect(to: clinic_path(conn, :index))
end

plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

defp authenticate(conn, _opts) do
  if conn.assigns.current_user do
  conn
```

```
    else
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Gender_controller.ex

```
defmodule Cmsv1.GenderController do
 use Cmsv1.Web, :controller

 alias Cmsv1.Gender

 def index(conn, _params) do
   genders = Repo.all(Gender)
   render(conn, "index.html", genders: genders)
 end

 def new(conn, _params) do
   changeset = Gender.changeset(%Gender{})
   render(conn, "new.html", changeset: changeset)
 end

 def create(conn, %{"gender" => gender_params}) do
   changeset = Gender.changeset(%Gender{}, gender_params)

   case Repo.insert(changeset) do
     {:ok, _gender} ->
       conn
       |> put_flash(:info, "Gender created successfully.")
       |> redirect(to: gender_path(conn, :index))
     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset)
   end
 end

 def show(conn, %{"id" => id}) do
   gender = Repo.get!(Gender, id)
   render(conn, "show.html", gender: gender)
 end
```

```
def edit(conn, %{"id" => id}) do
  gender = Repo.get!(Gender, id)
  changeset = Gender.changeset(gender)
  render(conn, "edit.html", gender: gender, changeset: changeset)
end

def update(conn, %{"id" => id, "gender" => gender_params}) do
  gender = Repo.get!(Gender, id)
  changeset = Gender.changeset(gender, gender_params)

  case Repo.update(changeset) do
    {:ok, gender} ->
      conn
      |> put_flash(:info, "Gender updated successfully.")
      |> redirect(to: gender_path(conn, :show, gender))
    {:error, changeset} ->
      render(conn, "edit.html", gender: gender, changeset: changeset)
  end
end

def delete(conn, %{"id" => id}) do
  gender = Repo.get!(Gender, id)

  # Here we use delete! (with a bang) because we expect
  # it to always work (and if it does not, it will raise).
  Repo.delete!(gender)

  conn
  |> put_flash(:info, "Gender deleted successfully.")
  |> redirect(to: gender_path(conn, :index))
end

plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

defp authenticate(conn, _opts) do
  if conn.assigns.current_user do
  conn
```

```
    else
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Gp_controller.ex

```
defmodule Cmsv1.GPController do
 use Cmsv1.Web, :controller

 alias Cmsv1.GP

 def index(conn, _params) do
   gps = Repo.all(GP)
   render(conn, "index.html", gps: gps)
 end

 def new(conn, _params) do
   changeset = GP.changeset(%GP{})
   render(conn, "new.html", changeset: changeset)
 end

 def create(conn, %{"gp" => gp_params}) do
   changeset = GP.changeset(%GP{}, gp_params)

   case Repo.insert(changeset) do
     {:ok, _gp} ->
       conn
       |> put_flash(:info, "Gp created successfully.")
       |> redirect(to: gp_path(conn, :index))
     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset)
   end
 end

 def show(conn, %{"id" => id}) do
   gp = Repo.get!(GP, id)
   render(conn, "show.html", gp: gp)
 end
```

```elixir
def edit(conn, %{"id" => id}) do
  gp = Repo.get!(GP, id)
  changeset = GP.changeset(gp)
  render(conn, "edit.html", gp: gp, changeset: changeset)
end

def update(conn, %{"id" => id, "gp" => gp_params}) do
  gp = Repo.get!(GP, id)
  changeset = GP.changeset(gp, gp_params)

  case Repo.update(changeset) do
    {:ok, gp} ->
      conn
      |> put_flash(:info, "Gp updated successfully.")
      |> redirect(to: gp_path(conn, :show, gp))
    {:error, changeset} ->
      render(conn, "edit.html", gp: gp, changeset: changeset)
  end
end

def delete(conn, %{"id" => id}) do
  gp = Repo.get!(GP, id)

  # Here we use delete! (with a bang) because we expect
  # it to always work (and if it does not, it will raise).
  Repo.delete!(gp)

  conn
  |> put_flash(:info, "Gp deleted successfully.")
  |> redirect(to: gp_path(conn, :index))
end

plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

defp authenticate(conn, _opts) do
  if conn.assigns.current_user do
  conn
  else
```

```
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Inactivity_controller

```
defmodule Cmsv1.InactivityController do
 use Cmsv1.Web, :controller

 alias Cmsv1.Inactivity

 def index(conn, _params) do
   reasons = Repo.all(Inactivity)
   render(conn, "index.html", reasons: reasons)
 end

 def new(conn, _params) do
   changeset = Inactivity.changeset(%Inactivity{})
   render(conn, "new.html", changeset: changeset)
 end

 def create(conn, %{"inactivity" => inactivity_params}) do
   changeset = Inactivity.changeset(%Inactivity{}, inactivity_params)

   case Repo.insert(changeset) do
     {:ok, _inactivity} ->
       conn
       |> put_flash(:info, "Inactivity created successfully.")
       |> redirect(to: inactivity_path(conn, :index))
     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset)
   end
 end

 def show(conn, %{"id" => id}) do
   inactivity = Repo.get!(Inactivity, id)
   render(conn, "show.html", inactivity: inactivity)
 end
```

```
def edit(conn, %{"id" => id}) do
  inactivity = Repo.get!(Inactivity, id)
  changeset = Inactivity.changeset(inactivity)
  render(conn, "edit.html", inactivity: inactivity, changeset: changeset)
end

def update(conn, %{"id" => id, "inactivity" => inactivity_params}) do
  inactivity = Repo.get!(Inactivity, id)
  changeset = Inactivity.changeset(inactivity, inactivity_params)

  case Repo.update(changeset) do
    {:ok, inactivity} ->
      conn
      |> put_flash(:info, "Inactivity updated successfully.")
      |> redirect(to: inactivity_path(conn, :show, inactivity))
    {:error, changeset} ->
      render(conn, "edit.html", inactivity: inactivity, changeset: changeset)
  end
end

def delete(conn, %{"id" => id}) do
  inactivity = Repo.get!(Inactivity, id)

  # Here we use delete! (with a bang) because we expect
  # it to always work (and if it does not, it will raise).
  Repo.delete!(inactivity)

  conn
  |> put_flash(:info, "Inactivity deleted successfully.")
  |> redirect(to: inactivity_path(conn, :index))
end

plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

defp authenticate(conn, _opts) do
  if conn.assigns.current_user do
  conn
```

```
    else
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Page_controller.ex

```
defmodule Cmsv1.PageController do
 use Cmsv1.Web, :controller

 def index(conn, _params) do
   render conn, "index.html"
 end
end
```

Patient_controller.ex

```
defmodule Cmsv1.PatientController do
 use Cmsv1.Web, :controller
 use Drab.Controller

  plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]
 import Ecto.Changeset
 import Ecto.Date

 alias Cmsv1.Patient
 alias Cmsv1.CDoctor
 alias Cmsv1.GP
 alias Cmsv1.Pharmacy
 alias Cmsv1.Inactivity
 alias Cmsv1.Gender
 alias Cmsv1.Relationship
 alias Cmsv1.VaccBrand

 def index(conn, _params) do
   patients = Repo.all(Patient)
   render(conn, "index.html", patients: patients)

 end

 def new(conn, _params) do
   changeset = Patient.changeset(%Patient{})

   doctors = Repo.all(CDoctor) |> Enum.map(&{&1.name, &1.cdoctor_id}) |>
Enum.into(%{})
   gps = Repo.all(GP) |> Enum.map(&{&1.name, &1.gp_id}) |> Enum.into(%{})
   pharms = Repo.all(Pharmacy) |> Enum.map(&{&1.name<>" , "<>&1.address,
&1.pharm_id}) |> Enum.into(%{})
   inactivity = Repo.all(Inactivity) |> Enum.map(&{&1.reason, &1.reason}) |>
Enum.into(%{})
   relations = Repo.all(Relationship) |> Enum.map(&{&1.relationship,
&1.relationship}) |> Enum.into(%{})
   genders = Repo.all(Gender) |> Enum.map(&{&1.gender, &1.gender}) |>
Enum.into(%{})
```

```
    render(conn, "new.html", changeset: changeset, doctors: doctors, gps: gps,
pharms: pharms, inactivity: inactivity, genders: genders, relations: relations)
  end

 def create(conn, %{"patient" => patient_params}) do
    changeset = Patient.changeset(%Patient{}, patient_params)

    dob = AgeCalc.age(get_change(changeset, :date_of_birth))
    changeset = change(changeset, %{age: dob})
    clinic = get_session(conn, :clinic_id)
    changeset = change(changeset, %{clinic_id: clinic})

    doctors = Repo.all(CDoctor) |> Enum.map(&{&1.name, &1.cdoctor_id}) |>
Enum.into(%{})
    gps = Repo.all(GP) |> Enum.map(&{&1.name, &1.gp_id}) |> Enum.into(%{})
    pharms = Repo.all(Pharmacy) |> Enum.map(&{&1.name<>" , "<>&1.address,
&1.pharm_id}) |> Enum.into(%{})
    inactivity = Repo.all(Inactivity) |> Enum.map(&{&1.reason, &1.reason}) |>
Enum.into(%{})
    relations = Repo.all(Relationship) |> Enum.map(&{&1.relationship,
&1.relationship}) |> Enum.into(%{})
    genders = Repo.all(Gender) |> Enum.map(&{&1.gender, &1.gender}) |>
Enum.into(%{})


    case Repo.insert(changeset) do
      {:ok, _patient} ->
        conn
        |> put_flash(:info, "Patient created successfully.")
        |> redirect(to: patient_path(conn, :index))
      {:error, changeset} ->
    render(conn, "new.html", changeset: changeset, doctors: doctors, gps: gps,
pharms: pharms, inactivity: inactivity, genders: genders, relations: relations)
    end
 end
  def show(conn, %{"id" => id}) do
    patient = Repo.get!(Patient, id)
    doctor = Repo.get!(CDoctor, patient.cdoctor_id)
```

```
    gp = Repo.get!(GP, patient.gp_id)
    pharmacy = Repo.get!(Pharmacy, patient.pharm_id)
    render(conn, "show.html", patient: patient, doctor: doctor, gp: gp, pharmacy:
pharmacy)
  end

  def show(conn, %{"patient_id" => id, "Type" => t}) do
    patient = Repo.get!(Patient, id)
    doctor = Repo.get!(CDoctor, patient.cdoctor_id)
    gp = Repo.get!(GP, patient.gp_id)
    pharmacy = Repo.get!(Pharmacy, patient.pharm_id)
    render(conn, "show.html", patient: patient, doctor: doctor, gp: gp, pharmacy:
pharmacy)
  end

  def edit(conn, %{"id" => id}) do
    patient = Repo.get!(Patient, id)

    doctors = Repo.all(CDoctor) |> Enum.map(&{&1.name, &1.cdoctor_id}) |>
Enum.into(%{})
    gps = Repo.all(GP) |> Enum.map(&{&1.name, &1.gp_id}) |> Enum.into(%{})
    pharms = Repo.all(Pharmacy) |> Enum.map(&{&1.name, &1.pharm_id}) |>
Enum.into(%{})
    inactivity = Repo.all(Inactivity) |> Enum.map(&{&1.reason, &1.reason}) |>
Enum.into(%{})
    relations = Repo.all(Relationship) |> Enum.map(&{&1.relationship,
&1.relationship}) |> Enum.into(%{})
    genders = Repo.all(Gender) |> Enum.map(&{&1.gender, &1.gender}) |>
Enum.into(%{})

    changeset = Patient.changeset(patient)
    render(conn, "edit.html", patient: patient, changeset: changeset, doctors:
doctors, gps: gps, pharms: pharms, inactivity: inactivity, genders: genders,
relations: relations)
  end

  def update(conn, %{"id" => id, "patient" => patient_params}) do
    patient = Repo.get!(Patient, id)
```

```
    changeset = Patient.changeset(patient, patient_params)

    case Repo.update(changeset) do
      {:ok, patient} ->
        conn
        |> put_flash(:info, "Patient updated successfully.")
        |> redirect(to: patient_path(conn, :show, patient))
      {:error, changeset} ->
        render(conn, "edit.html", patient: patient, changeset: changeset)
    end
  end

  def delete(conn, %{"id" => id}) do
    patient = Repo.get!(Patient, id)
    Repo.delete!(patient)

    conn
    |> put_flash(:info, "Patient deleted successfully.")
    |> redirect(to: patient_path(conn, :index))
  end

  defp authenticate(conn, _opts) do
    if conn.assigns.current_user do
    conn
    else
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Pharmacy_controller.ex

```
defmodule Cmsv1.PharmacyController do
 use Cmsv1.Web, :controller

 alias Cmsv1.Pharmacy

 def index(conn, _params) do
   pharms = Repo.all(Pharmacy)
   render(conn, "index.html", pharms: pharms)
 end

 def new(conn, _params) do
   changeset = Pharmacy.changeset(%Pharmacy{})
   render(conn, "new.html", changeset: changeset)
 end

 def create(conn, %{"pharmacy" => pharmacy_params}) do
   changeset = Pharmacy.changeset(%Pharmacy{}, pharmacy_params)

   case Repo.insert(changeset) do
     {:ok, _pharmacy} ->
       conn
       |> put_flash(:info, "Pharmacy created successfully.")
       |> redirect(to: pharmacy_path(conn, :index))
     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset)
   end
 end

 def show(conn, %{"id" => id}) do
   pharmacy = Repo.get!(Pharmacy, id)
   render(conn, "show.html", pharmacy: pharmacy)
 end
```

```elixir
def edit(conn, %{"id" => id}) do
  pharmacy = Repo.get!(Pharmacy, id)
  changeset = Pharmacy.changeset(pharmacy)
  render(conn, "edit.html", pharmacy: pharmacy, changeset: changeset)
end

def update(conn, %{"id" => id, "pharmacy" => pharmacy_params}) do
  pharmacy = Repo.get!(Pharmacy, id)
  changeset = Pharmacy.changeset(pharmacy, pharmacy_params)

  case Repo.update(changeset) do
    {:ok, pharmacy} ->
      conn
      |> put_flash(:info, "Pharmacy updated successfully.")
      |> redirect(to: pharmacy_path(conn, :show, pharmacy))
    {:error, changeset} ->
      render(conn, "edit.html", pharmacy: pharmacy, changeset: changeset)
  end
end

def delete(conn, %{"id" => id}) do
  pharmacy = Repo.get!(Pharmacy, id)

  # Here we use delete! (with a bang) because we expect
  # it to always work (and if it does not, it will raise).
  Repo.delete!(pharmacy)

  conn
  |> put_flash(:info, "Pharmacy deleted successfully.")
  |> redirect(to: pharmacy_path(conn, :index))
end

plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

defp authenticate(conn, _opts) do
  if conn.assigns.current_user do
  conn
```

```
    else
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Phlebotomy_controller.ex

```
defmodule Cmsv1.PhlebotomyController do
 use Cmsv1.Web, :controller

 import Ecto.Changeset

 alias Cmsv1.Phlebotomy
 alias Cmsv1.Patient

 def index(conn, _params) do
   phleb = Repo.all(Phlebotomy)
   patients = Repo.all(Patient)
   render(conn, "index.html", phleb: phleb, patients: patients)
 end

 def new(conn, _params) do
   changeset = Phlebotomy.changeset(%Phlebotomy{})

   patients = Repo.all(Patient) |> Enum.map(&{&1.fname<>" "<>&1.lname,
&1.patient_id}) |> Enum.into(%{})
   render(conn, "new.html", changeset: changeset, patients: patients)
 end

 def create(conn, %{"phlebotomy" => phlebotomy_params}) do
   changeset = Phlebotomy.changeset(%Phlebotomy{}, phlebotomy_params)

   patients = Repo.all(Patient) |> Enum.map(&{&1.fname, &1.patient_id}) |>
Enum.into(%{})

   clinic = get_session(conn, :clinic_id)
   changeset = change(changeset, %{clinic_id: clinic})

   case Repo.insert(changeset) do
```

```
      {:ok, _phlebotomy} ->
        conn
        |> put_flash(:info, "Phlebotomy created successfully.")
        |> redirect(to: phlebotomy_path(conn, :index))
      {:error, changeset} ->
        render(conn, "new.html", changeset: changeset, patients: patients)
    end
  end

  def show(conn, %{"id" => id}) do
    phlebotomy = Repo.get!(Phlebotomy, id)
    patients = Repo.all(Patient)
    render(conn, "show.html", phlebotomy: phlebotomy, patients: patients)
  end

  def edit(conn, %{"id" => id}) do
    phlebotomy = Repo.get!(Phlebotomy, id)
    patients = Repo.all(Patient) |> Enum.map(&{&1.fname<>" "<>&1.lname,
&1.patient_id}) |> Enum.into(%{})
    changeset = Phlebotomy.changeset(phlebotomy)
    render(conn, "edit.html", phlebotomy: phlebotomy, changeset: changeset,
patients: patients)
  end

  def update(conn, %{"id" => id, "phlebotomy" => phlebotomy_params}) do
    phlebotomy = Repo.get!(Phlebotomy, id)
    changeset = Phlebotomy.changeset(phlebotomy, phlebotomy_params)

    case Repo.update(changeset) do
      {:ok, phlebotomy} ->
        conn
        |> put_flash(:info, "Phlebotomy updated successfully.")
        |> redirect(to: phlebotomy_path(conn, :show, phlebotomy))
      {:error, changeset} ->
        render(conn, "edit.html", phlebotomy: phlebotomy, changeset: changeset)
    end
  end
```

```
 def delete(conn, %{"id" => id}) do
   phlebotomy = Repo.get!(Phlebotomy, id)

   # Here we use delete! (with a bang) because we expect
   # it to always work (and if it does not, it will raise).
   Repo.delete!(phlebotomy)

   conn
   |> put_flash(:info, "Phlebotomy deleted successfully.")
   |> redirect(to: phlebotomy_path(conn, :index))
 end

 plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

 defp authenticate(conn, _opts) do
   if conn.assigns.current_user do
   conn
   else
   conn
   |> put_flash(:error, "You must be logged in to access that page")
   |> redirect(to: session_path(conn, :new))
   |> halt()
   end
   end
end
```

Relationship_controller.ex

```
defmodule Cmsv1.RelationshipController do
 use Cmsv1.Web, :controller

 alias Cmsv1.Relationship

 def index(conn, _params) do
   relationships = Repo.all(Relationship)
   render(conn, "index.html", relationships: relationships)
 end

 def new(conn, _params) do
   changeset = Relationship.changeset(%Relationship{})
   render(conn, "new.html", changeset: changeset)
 end

 def create(conn, %{"relationship" => relationship_params}) do
   changeset = Relationship.changeset(%Relationship{}, relationship_params)

   case Repo.insert(changeset) do
     {:ok, _relationship} ->
       conn
       |> put_flash(:info, "Relationship created successfully.")
       |> redirect(to: relationship_path(conn, :index))
     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset)
   end
 end

 def show(conn, %{"id" => id}) do
   relationship = Repo.get!(Relationship, id)
   render(conn, "show.html", relationship: relationship)
 end
```

```
def edit(conn, %{"id" => id}) do
  relationship = Repo.get!(Relationship, id)
  changeset = Relationship.changeset(relationship)
  render(conn, "edit.html", relationship: relationship, changeset: changeset)
end

def update(conn, %{"id" => id, "relationship" => relationship_params}) do
  relationship = Repo.get!(Relationship, id)
  changeset = Relationship.changeset(relationship, relationship_params)

  case Repo.update(changeset) do
    {:ok, relationship} ->
      conn
      |> put_flash(:info, "Relationship updated successfully.")
      |> redirect(to: relationship_path(conn, :show, relationship))
    {:error, changeset} ->
      render(conn, "edit.html", relationship: relationship, changeset: changeset)
  end
end

def delete(conn, %{"id" => id}) do
  relationship = Repo.get!(Relationship, id)

  # Here we use delete! (with a bang) because we expect
  # it to always work (and if it does not, it will raise).
  Repo.delete!(relationship)

  conn
  |> put_flash(:info, "Relationship deleted successfully.")
  |> redirect(to: relationship_path(conn, :index))
end

plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

defp authenticate(conn, _opts) do
  if conn.assigns.current_user do
  conn
```

```
    else
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Report_controller.ex

```
defmodule Cmsv1.ReportController do
   use Cmsv1.Web, :controller

   alias Cmsv1.GP
   alias Cmsv1.Patient

   def index(conn, _params) do

     patients = Repo.all(Patient) |> Repo.preload([:gps, :pharms, :cdoctors])
     render conn, "index.html", patients: patients
   end
 end
```

Session_controller.ex

```
defmodule Cmsv1.SessionController do
   use Cmsv1.Web, :controller

   alias Cmsv1.Repo
   def new(conn, _) do
       render conn, "new.html"
   end

   # create session (login)
   def create(conn, %{"session" => %{"username" => user, "password" => pass}}) do
       case Cmsv1.Auth.login_by_username_and_pass(conn, user, pass, repo: Repo) do

           {:ok, conn} ->
               user = get_session(conn, :level)
               if user == "admin" do
                   IO.inspect("IS ADMIN")
                   conn
                   |> put_flash(:info, "Welcome back!")
                   |> redirect(to: user_path(conn, :index))
               else
                   IO.inspect("IS STANDARD")
                   conn
                   |> put_flash(:info, "Welcome back!")
                   |> redirect(to: patient_path(conn, :index))
               end

           {:error, _reason, conn} ->
               conn
               |> put_flash(:error, "Invalid username/password combination")
               |> render("new.html")
       end
   end
```

```
    # delete session (logout)
    def delete(conn, _) do
        conn
        |> Cmsv1.Auth.logout()
        |> redirect(to: session_path(conn, :new))
        end
end
```

User_controller.ex

```
defmodule Cmsv1.UserController do
   use Cmsv1.Web, :controller

   #plug :authenticate when action in [:index, :show]

   alias Cmsv1.User
   alias Cmsv1.Clinic

   def index(conn, _params) do
       users = Repo.all(Cmsv1.User) |> Repo.preload([:clinics])
       render conn, "index.html", users: users
       end

   def show(conn, %{"id" => id}) do
       user = Repo.get(Cmsv1.User, id)
       render conn, "show.html", user: user
   end

   def new(conn, _params) do
       changeset = User.changeset(%User{})
       clinics = Repo.all(Clinic) |> Enum.map(&{&1.name, &1.clinic_id}) |>
Enum.into(%{})
       render conn, "new.html", changeset: changeset, clinics: clinics
   end

   def create(conn, %{"user" => user_params}) do
       changeset = User.registration_changeset(%User{}, user_params)

       case Repo.insert(changeset) do
           {:ok, user} ->
               conn
               |> put_flash(:info, "#{user.name} created!")
               |> redirect(to: user_path(conn, :index))
```

```
            {:error, changeset} ->
                render(conn, "new.html", changeset: changeset)
        end
    end

    defp authenticate(conn, _opts) do
        if conn.assigns.current_user do
        conn
        else
        conn
        |> put_flash(:error, "You must be logged in to access that page")
        |> redirect(to: session_path(conn, :new))
        |> halt()
        end
        end
end
```

Vacc_brand_controller.ex

```
defmodule Cmsv1.VaccBrandController do
 use Cmsv1.Web, :controller

 alias Cmsv1.VaccBrand

 def index(conn, _params) do
   vaccbrands = Repo.all(VaccBrand)
   render(conn, "index.html", vaccbrands: vaccbrands)
 end

 def new(conn, _params) do
   changeset = VaccBrand.changeset(%VaccBrand{})
   render(conn, "new.html", changeset: changeset)
 end

 def create(conn, %{"vacc_brand" => vacc_brand_params}) do
   changeset = VaccBrand.changeset(%VaccBrand{}, vacc_brand_params)

   case Repo.insert(changeset) do
     {:ok, _vacc_brand} ->
       conn
       |> put_flash(:info, "Vacc brand created successfully.")
       |> redirect(to: vacc_brand_path(conn, :index))
     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset)
   end
 end

 def show(conn, %{"id" => id}) do
   vacc_brand = Repo.get!(VaccBrand, id)
   render(conn, "show.html", vacc_brand: vacc_brand)
 end
```

```
def edit(conn, %{"id" => id}) do
  vacc_brand = Repo.get!(VaccBrand, id)
  changeset = VaccBrand.changeset(vacc_brand)
  render(conn, "edit.html", vacc_brand: vacc_brand, changeset: changeset)
end

def update(conn, %{"id" => id, "vacc_brand" => vacc_brand_params}) do
  vacc_brand = Repo.get!(VaccBrand, id)
  changeset = VaccBrand.changeset(vacc_brand, vacc_brand_params)

  case Repo.update(changeset) do
    {:ok, vacc_brand} ->
      conn
      |> put_flash(:info, "Vacc brand updated successfully.")
      |> redirect(to: vacc_brand_path(conn, :show, vacc_brand))
    {:error, changeset} ->
      render(conn, "edit.html", vacc_brand: vacc_brand, changeset: changeset)
  end
end

def delete(conn, %{"id" => id}) do
  vacc_brand = Repo.get!(VaccBrand, id)

  # Here we use delete! (with a bang) because we expect
  # it to always work (and if it does not, it will raise).
  Repo.delete!(vacc_brand)

  conn
  |> put_flash(:info, "Vacc brand deleted successfully.")
  |> redirect(to: vacc_brand_path(conn, :index))
end

plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

defp authenticate(conn, _opts) do
  if conn.assigns.current_user do
  conn
  else
```

```
    conn
    |> put_flash(:error, "You must be logged in to access that page")
    |> redirect(to: session_path(conn, :new))
    |> halt()
    end
    end
end
```

Vaccinations_controller.ex

```
defmodule Cmsv1.VaccinationsController do
 use Cmsv1.Web, :controller

 import Ecto.Changeset
  alias Cmsv1.Vaccinations
 alias Cmsv1.Patient
 alias Cmsv1.VaccBrand

 def index(conn, _params) do
   vaccs = Repo.all(Vaccinations)
   patients = Repo.all(Patient)
   render(conn, "index.html", vaccs: vaccs, patients: patients)
 end

 def new(conn, _params) do
   changeset = Vaccinations.changeset(%Vaccinations{})
   patients = Repo.all(Patient) |> Enum.map(&{&1.fname<>" "<>&1.lname,
&1.patient_id}) |> Enum.into(%{})
   brands = Repo.all(VaccBrand) |> Enum.map(&{&1.vaccbrand, &1.vaccbrand}) |>
Enum.into(%{})
   render(conn, "new.html", changeset: changeset, patients: patients, brands:
brands)
 end

 def create(conn, %{"vaccinations" => vaccinations_params}) do
   changeset = Vaccinations.changeset(%Vaccinations{}, vaccinations_params)
   patients = Repo.all(Patient) |> Enum.map(&{&1.fname, &1.patient_id}) |>
Enum.into(%{})
   brands = Repo.all(VaccBrand) |> Enum.map(&{&1.vaccbrand, &1.vaccbrand}) |>
Enum.into(%{})

   clinic = get_session(conn, :clinic_id)
   changeset = change(changeset, %{clinic_id: clinic})
```

```
   case Repo.insert(changeset) do
     {:ok, _vaccinations} ->
       if (get_field(changeset, :revacc_type) == "Full course") do
         conn
         |> put_flash(:info, "Vaccinations created successfully. (Full course
selected, please enter a new vaccination record)")
         |> redirect(to: vaccinations_path(conn, :new))
       else
         conn
         |> put_flash(:info, "Vaccinations created successfully.")
         |> redirect(to: vaccinations_path(conn, :index))
       end

     {:error, changeset} ->
       render(conn, "new.html", changeset: changeset, patients: patients, brands:
brands)
   end
 end

 def show(conn, %{"id" => id}) do
   vaccinations = Repo.get!(Vaccinations, id)
   patients = Repo.all(Patient)
   render(conn, "show.html", vaccinations: vaccinations, patients: patients)
 end

 def edit(conn, %{"id" => id}) do
   vaccinations = Repo.get!(Vaccinations, id)
   patients = Repo.all(Patient) |> Enum.map(&{&1.fname<>" "<>&1.lname,
&1.patient_id}) |> Enum.into(%{})
   brands = Repo.all(VaccBrand) |> Enum.map(&{&1.vaccbrand, &1.vaccbrand}) |>
Enum.into(%{})
   changeset = Vaccinations.changeset(vaccinations)
   render(conn, "edit.html", vaccinations: vaccinations, changeset: changeset,
patients: patients, brands: brands)
 end
```

```
  def update(conn, %{"id" => id, "vaccinations" => vaccinations_params}) do
    vaccinations = Repo.get!(Vaccinations, id)
    changeset = Vaccinations.changeset(vaccinations, vaccinations_params)

    case Repo.update(changeset) do
      {:ok, vaccinations} ->
        conn
        |> put_flash(:info, "Vaccinations updated successfully.")
        |> redirect(to: vaccinations_path(conn, :show, vaccinations))
      {:error, changeset} ->
        render(conn, "edit.html", vaccinations: vaccinations, changeset: changeset)
    end
  end

  def delete(conn, %{"id" => id}) do
    vaccinations = Repo.get!(Vaccinations, id)
    Repo.delete!(vaccinations)
    conn
    |> put_flash(:info, "Vaccinations deleted successfully.")
    |> redirect(to: vaccinations_path(conn, :index))
  end

  plug :authenticate when action in [:index, :show, :new, :edit, :update, :delete]

  defp authenticate(conn, _opts) do
    if conn.assigns.current_user do
      conn
    else
      conn
      |> put_flash(:error, "You must be logged in to access that page")
      |> redirect(to: session_path(conn, :new))
      |> halt()
    end
    end
end
```

Models

Cdoctor.ex

```
defmodule Cmsv1.CDoctor do
 use Cmsv1.Web, :model
 @primary_key {:cdoctor_id, :binary_id, autogenerate: true}
 @derive {Phoenix.Param, key: :cdoctor_id}

 schema "cdoctors" do
   field :name, :string
   field :address, :string
   field :mobile_number, :string
   field :landline_number, :string
   field :fax, :string
   field :email, :string
   field :training_level, :string
   has_one :patients, Cmsv1.Patient

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:name, :address, :mobile_number, :landline_number, :fax,
:email, :training_level])
   |> validate_required([:name, :address, :mobile_number, :landline_number, :fax,
:email, :training_level])
 end
end
```

Clinic.ex

```
defmodule Cmsv1.Clinic do
 use Cmsv1.Web, :model

 @primary_key {:clinic_id, :binary_id, autogenerate: true}
 @derive {Phoenix.Param, key: :clinic_id}

 schema "clinics" do
   field :name, :string
   field :address, :string
   has_one :patients, Cmsv1.Patient
   has_one :vaccs, Cmsv1.Vaccination
   has_one :phleb, Cmsv1.Phlebotomy
   has_one :clinics, Cmsv1.Clinic


   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:name, :address])
   |> validate_required([:name, :address])
 end
end
```

Gender.ex

```
defmodule Cmsv1.Gender do
 use Cmsv1.Web, :model

 schema "genders" do
   field :gender, :string

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:gender])
   |> validate_required([:gender])
 end
end
```

Gp.ex

```
defmodule Cmsv1.GP do
 use Cmsv1.Web, :model

 @primary_key {:gp_id, :binary_id, autogenerate: true}
 @derive {Phoenix.Param, key: :gp_id}

 schema "gps" do
   field :name, :string
   field :address, :string
   field :mobile_number, :string
   field :landline_number, :string
   field :fax, :string
   field :email, :string
   field :training_level, :string
   has_one :patients, Cmsv1.Patient

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:name, :address, :mobile_number, :landline_number, :fax,
:email, :training_level])
   |> validate_required([:name, :address, :mobile_number, :landline_number, :fax,
:email, :training_level])
 end
end
```

Inactivity.ex

```
defmodule Cmsv1.Inactivity do
 use Cmsv1.Web, :model

 schema "reasons" do
   field :reason, :string

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:reason])
   |> validate_required([:reason])
 end
end
```

Patient.ex

```
defmodule Cmsv1.Patient do
 use Cmsv1.Web, :model
 import Ecto.Changeset
  @primary_key {:patient_id, :binary_id, autogenerate: true}
 @derive {Phoenix.Param, key: :patient_id}
  schema "patients" do
   field :ph_number, Cloak.EncryptedBinaryField
   field :fname, Cloak.EncryptedBinaryField
   field :lname, Cloak.EncryptedBinaryField
   field :street, Cloak.EncryptedBinaryField
   field :towncity, Cloak.EncryptedBinaryField
   field :county, Cloak.EncryptedBinaryField
   field :eircode, Cloak.EncryptedBinaryField
   field :ppsn, Cloak.EncryptedBinaryField
   field :date_of_birth, Ecto.Date
   field :gender, :string
   field :medical_card_present, :boolean, default: false
   field :medical_card_number, Cloak.EncryptedBinaryField
   field :medical_card_expiry, Ecto.Date
   field :mobile_number, :string
   field :landline_number, :string
   field :nok_rel, :binary
   field :nok_name, :string
   field :nok_address, :string
   field :nok_mobile_number, :string
   field :nok_landline_number, :string
   field :active, :boolean, default: false
   field :active_details, :string
   field :age, :integer
   field :transfer_status, :boolean, default: false
   belongs_to :clinics, Cmsv1.Clinic, foreign_key: :clinic_id, type: :binary_id,
references: :clinic_id
   belongs_to :gps, Cmsv1.GP, foreign_key: :gp_id, type: :binary_id, references:
```

```
:gp_id
   belongs_to :cdoctors, Cmsv1.CDoctor, foreign_key: :cdoctor_id, type: :binary_id,
references: :cdoctor_id
   belongs_to :pharms, Cmsv1.Pharmacy, foreign_key: :pharm_id, type: :binary_id,
references: :pharm_id
   has_many :phleb, Cmsv1.Phlebotomy
   has_many :vaccs, Cmsv1.Vaccinations


   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do

   struct
   |> cast(params, [:ph_number, :fname, :lname, :street, :towncity, :county,
:eircode, :ppsn, :date_of_birth, :gender, :medical_card_present,
:medical_card_number, :medical_card_expiry, :mobile_number, :landline_number,
:nok_rel, :nok_name, :nok_address, :nok_mobile_number, :nok_landline_number,
:active, :active_details, :cdoctor_id, :gp_id, :pharm_id, :transfer_status,
:clinic_id])
   |> validate_required([:ph_number, :fname, :lname, :ppsn, :date_of_birth,
:gender, :medical_card_present, :mobile_number, :landline_number, :nok_rel,
:nok_name, :nok_address, :nok_mobile_number, :nok_landline_number, :active,
:cdoctor_id, :gp_id, :pharm_id])
   |> foreign_key_constraint(:gp_id)
   |> foreign_key_constraint(:pharm_id)
   |> foreign_key_constraint(:cdoctor_id)
   |> foreign_key_constraint(:clinic_id)

 end

end
```

Pharmacy.ex

```
defmodule Cmsv1.Pharmacy do
 use Cmsv1.Web, :model

 @primary_key {:pharm_id, :binary_id, autogenerate: true}
 @derive {Phoenix.Param, key: :pharm_id}

 schema "pharms" do
   field :name, :string
   field :address, :string
   field :mobile_number, :string
   field :landline_number, :string
   field :fax, :string
   field :email, :string
   field :contact_person, :string
   field :contact_person_number, :string
   has_one :patients, Cmsv1.Patient

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:name, :address, :mobile_number, :landline_number, :fax,
:email, :contact_person, :contact_person_number])
   |> validate_required([:name, :address, :mobile_number, :landline_number, :fax,
:email, :contact_person, :contact_person_number])
 end
end
```

Phlebotomy.ex

```
defmodule Cmsv1.Phlebotomy do
 use Cmsv1.Web, :model

 @primary_key {:phleb_id, :binary_id, autogenerate: true}
 @derive {Phoenix.Param, key: :phleb_id}

 schema "phleb" do
   field :hiv_status, :boolean, default: false
   field :hiv_date, Ecto.Date
   field :hepc_status, :boolean, default: false
   field :hepc_date, Ecto.Date
   field :hepa_status, :boolean, default: false
   field :hepa_date, Ecto.Date
   field :hepb_status, :boolean, default: false
   field :hepb_date, Ecto.Date
   field :ref_status, :boolean, default: false
   field :ref_date, Ecto.Date
   field :ref_to, :string
   belongs_to :clinics, Cmsv1.Clinic, foreign_key: :clinic_id, type: :binary_id,
references: :clinic_id
   belongs_to :patients, Cmsv1.Patient, foreign_key: :patient_id, type: :binary_id,
references: :patient_id

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:patient_id, :hiv_status, :hiv_date, :hepc_status, :hepc_date,
:hepa_status, :hepa_date, :hepb_status, :hepb_date, :ref_status, :ref_date,
```

```
:ref_to, :clinic_id])
   |> validate_required([:hiv_status, :hiv_date, :hepc_status, :hepc_date,
:hepa_status, :hepa_date, :hepb_status, :hepb_date])
   |> foreign_key_constraint(:patient_id)
   |> foreign_key_constraint(:clinic_id)
 end
end
```

Relationship.ex

```
defmodule Cmsv1.Relationship do
 use Cmsv1.Web, :model

 schema "relationships" do
   field :relationship, :string

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:relationship])
   |> validate_required([:relationship])
 end
end
```

User.ex

```
defmodule Cmsv1.User do
   use Cmsv1.Web, :model
   schema "users" do
       field :name, :string
       field :username, :string
       field :password, :string, virtual: true
       field :password_hash, :string
       field :level, :string
       belongs_to :clinics, Cmsv1.Clinic, foreign_key: :clinic_id, type:
:binary_id, references: :clinic_id

       timestamps()
   end

   def changeset(model, params \\ %{}) do
       model
       |> cast(params, ~w(name username), [:clinic_id, :level])
       |> validate_length(:username, min: 1, max: 20)
   end

   def registration_changeset(model, params) do
       model
       |> changeset(params)
       |> cast(params, [:name, :username, :password, :clinic_id, :level])
       |> validate_length(:password, min: 6, max: 100)
       |> foreign_key_constraint(:clinic_id)
       |> put_pass_hash()
   end

   defp put_pass_hash(changeset) do
       case changeset do
       %Ecto.Changeset{valid?: true, changes: %{password: pass}} ->
       put_change(changeset, :password_hash, Comeonin.Bcrypt.hashpwsalt(pass))
```

```
        _ ->
        changeset
        end
        end
end
```

Vaccbrand.ex

```
defmodule Cmsv1.VaccBrand do
 use Cmsv1.Web, :model

 schema "vaccbrands" do
   field :vaccbrand, :string

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
   struct
   |> cast(params, [:vaccbrand])
   |> validate_required([:vaccbrand])
 end
end
```

Vaccinations.ex

```
defmodule Cmsv1.Vaccinations do
 use Cmsv1.Web, :model

 @primary_key {:vacc_id, :binary_id, autogenerate: true}
 @derive {Phoenix.Param, key: :vacc_id}

 schema "vaccs" do
   field :vacc_brand, :string
   field :dose1_status, :boolean, default: false
   field :dose1_date, Ecto.Date
   field :dose2_status, :boolean, default: false
   field :dose2_date, Ecto.Date
   field :dose3_status, :boolean, default: false
   field :dose3_date, Ecto.Date
   field :hbs_status, :boolean, default: false
   field :hbs_result, :string
   field :revacc_status, :boolean, default: false
   field :revacc_date, Ecto.Date
   field :revacc_type, :string
   field :booster_date, Ecto.Date
   field :booster_dose, :string
   belongs_to :clinics, Cmsv1.Clinic, foreign_key: :clinic_id, type: :binary_id,
references: :clinic_id
   belongs_to :patients, Cmsv1.Patient, foreign_key: :patient_id, type: :binary_id,
references: :patient_id

   timestamps()
 end

 @doc """
 Builds a changeset based on the `struct` and `params`.
 """
 def changeset(struct, params \\ %{}) do
```

```
    struct
    |> cast(params, [:patient_id, :vacc_brand, :dose1_status, :dose1_date,
:dose2_status, :dose2_date, :dose3_status, :dose3_date, :hbs_status, :hbs_result,
:revacc_status, :revacc_type, :booster_date, :booster_dose, :clinic_id])
    |> validate_required([:vacc_brand, :dose1_status, :dose2_status, :dose3_status,
:hbs_status, :revacc_status])
    |> foreign_key_constraint(:patient_id)
    |> foreign_key_constraint(:clinic_id)
  end
end
```

View (Sample)

Patient_view.ex

```
defmodule Cmsv1.PatientView do
 use Cmsv1.Web, :view
end
```

Template (Sample)

App.html.eex

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <meta name="description" content="">
   <meta name="author" content="">

   <title>Ardu Management System</title>
   <link rel="stylesheet" href="<%= static_path(@conn, "/css/app.css") %>">
 </head>

 <body class="bg">

 <div class="container-fluid full">
    <header class="header">
         <nav class="navbar navbar-default navbar-fixed-top navbar-inverse">

           <div class="container-fluid">
             <div class="navbar-header">
               <a class="navbar-brand" href="#">Ardu Management System</a>
             </div>

             <ul class="nav navbar-nav">

             <%= if @current_user do %>
               <%= if @current_user.level == "admin" do %>
                 <li class="links"><%= link "Users", to: user_path(@conn,
:index)%></li>
                 <li class="links"><%=link("Clinic Management", to: "/clinics",
method: "get")%></li>
```

```
                </ul>
              <% else %>
                <li class="links"><%=link("Patient", to: "/patients", method:
"get")%></li>
                <li class="links"><%= link "Report", to: report_path(@conn,
:index)%></li>
                </ul>
                <div class="dropdown">
            <button class="dropbtn">Clinic Entities</button>
            <div class="dropdown-content">
              <%=link("Pharmacy", to: "/pharms", method: "get")%>
              <%=link("General Practitoner", to: "/gps", method: "get")%>
              <%=link("Clinic Doctor", to: "/cdoctors", method: "get")%>

          </div>
        </div>
        <div class="dropdown">
            <button class="dropbtn">Clinical Measurements</button>
            <div class="dropdown-content">
              <%=link("Phlebotomy", to: "/phleb", method: "get")%>
              <%=link("Vaccination", to: "/vaccs", method: "get")%>
            </div>
        </div>
        <div class="dropdown">
            <button class="dropbtn">Config</button>
            <div class="dropdown-content">
              <%=link("Patient gender options", to: "/genders", method:
"get")%></li>
                <%=link("Patient relationship options", to: "/relationships",
method: "get")%></li>
                <%=link("Inactivity reason options", to: "/reasons", method:
"get")%></li>
                <%=link("Vaccination brands", to: "/vaccbrands", method:
"get")%></li>
            </div>
        </div>
    <% end %>
    <% end %>
```

```
        </nav>
      </header>


    </div>

    <ol class="breadcrumb text-right">
      <%= if @current_user do %>
        <li>
          Logged in as: <%= @current_user.username %>
        </li>
        <li>
          Access level: <%= @current_user.level %>
        </li>
        <li>
          <%= link "Log out", to: session_path(@conn, :delete, @current_user),
method: "delete" %>
        </li>
      <% else %>
        <li>No user currently logged in</li>
        <li><%= link "Log in", to: session_path(@conn, :new) %></li>
      <% end %>
    </ol>
    <h4 class="alert alert-info" role="alert"><%= get_flash(@conn, :info) %></h4>
    <h4 class="alert alert-danger" role="alert"><%= get_flash(@conn, :error) %></h4>

    <div class="px-4">
      <div class="panel panel-default">
        <div class="panel-body">
          <main role="main">
              <%= render @view_module, @view_template, assigns %>
            </main>
        </div>
      </div>
    </div>
```

```
    </div> <!-- /container -->
    <script src="<%= static_path(@conn, "/js/app.js") %>"></script>
    <%= Drab.Client.run(@conn) %>
  </body>
</html>
```

patient/index.html

```
<h2>Patient</h2>
<div class="alert alert-info">
     <p>To sort the table, click the heading of the column you want to sort by.
Click to cycle through ascending and descending order.</p>
 </div>

<table class="sortable table">
 <thead>
   <tr>
     <th style="text-align: center;" ><button type="button" class="btn
btn-default">Name</th>
     <th style="text-align: center;" ><button type="button" class="btn
btn-default">Ppsn</th>
     <th style="text-align: center;" ><button type="button" class="btn
btn-default">Date of birth</th>

     <th></th>
   </tr>
 </thead>
 <tbody>
<%= for patient <- @patients do %>
 <%= if @current_user.clinic_id == patient.clinic_id do %>
   <tr>
     <td style="text-align: center;"><%= patient.fname<>" "<>patient.lname %></td>
     <td style="text-align: center;"><%= patient.ppsn %></td>
     <td style="text-align: center;"><%= patient.date_of_birth %></td>

     <td class="text-right">
       <%= link "Show", to: patient_path(@conn, :show, patient), class: "btn
btn-primary btn" %>
       <%= link "Edit", to: patient_path(@conn, :edit, patient), class: "btn
btn-primary btn" %>
       </td>
```

```
    </tr>
  <% end %>
<% end %>
  </tbody>
</table>

<%= link "Add Patient", to: patient_path(@conn, :new), class: "btn btn-primary btn"
%>
<script
src="https://www.kryogenix.org/code/browser/sorttable/sorttable.js"></script>
```

patient/form.html

```
<%= form_for @changeset, @action, fn f -> %>
 <%= if @changeset.action do %>
   <div class="alert alert-danger">
     <p>Oops, something went wrong! Please check the errors below.</p>
   </div>
 <% end %>

 <div class="alert alert-info">
     <p>A valid GP, Clinic Doctor and Pharmacy must be present to add a new
patient.</p>
 </div>

 <div class="alert alert-warning">
     <p>If the patient's GP, Pharmacy or Clinic Doctor is not found in the system,
please follow these steps:</p>
     <ul>
       <li>Open a new tab or window, and navigate to the desired page (for example,
Pharmacy).</li>
       <li>Add the new record into the system.</li>
       <li>Return to this page, and click the refresh button below.</li>
     </ul></br>
     <p>Once you have started filling out this form, do not close the page or
navigate away, or the data on the form will be lost!</p>
 </div>
  <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">First Name:</span>
   <%= text_input f, :fname, class: "form-control" %>
   <%= error_tag f, :fname %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Last Name:</span>
   <%= text_input f, :lname, class: "form-control" %>
```

```
    <%= error_tag f, :lname %>
  </div></br>

  <div class="input-group">
    <span class="input-group-addon" id="basic-addon1">Street Name/Number:</span>
    <%= text_input f, :street, class: "form-control", placeholder: "Street and
Number, P.O. box" %><p></p>
    <%= error_tag f, :street %>
  </div></br>

  <div class="input-group">
    <span class="input-group-addon" id="basic-addon1">Town/City:</span>
    <%= text_input f, :towncity, class: "form-control" %>
    <%= error_tag f, :towncity %>
  </div></br>

  <div class="input-group">
    <span class="input-group-addon" id="basic-addon1">County:</span>
    <%= select f,
:county,["Antrim","Armagh","Carlow","Cavan","Clare","Cork","Derry","Donegal","Down"
,"Dublin","Fermanagh","Galway","Kerry","Kildare","Kilkenny","Laois","Leitrim","Lime
rick","Longford","Louth","Mayo","Meath","Monaghan","Offaly","Roscommon","Sligo","Ti
pperary","Tyrone","Waterford","Westmeath","Wexford","Wicklow"], selected: "Carlow"
, class: "form-control" %>
    <%= error_tag f, :county %>
  </div></br>

  <div class="input-group">
    <span class="input-group-addon" id="basic-addon1">Eircode:</span>
    <%= text_input f, :eircode, class: "form-control" %>
    <%= error_tag f, :eircode %>
  </div></br>

  <div class="input-group">
    <span class="input-group-addon" id="basic-addon1">Date of birth:</span>
    <%= date_select f, :date_of_birth, year: [options: 1900..2100], class:
"form-control", style: "font-size: 20px" %>
    <%= error_tag f, :date_of_birth %>
```

```
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Patient's Pharmacy:</span>
   <%= select f, :pharm_id, @pharms, prompt: "Choose a pharmacy", class:
"form-control"%>

   <span class="input-group-addon">
     <a class="badge badge-warning" drab-click="refresh_pharms">Refresh</a>
   </span>
   <%= error_tag f, :pharm_id %>

 </div></br>
  <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Patient's Clinic
Doctor:</span>
   <%= select f, :cdoctor_id, @doctors, prompt: "Choose a clinic doctor", class:
"form-control" %>
   <span class="input-group-addon">
     <a class="badge badge-warning" drab-click="refresh_doctors">Refresh</a>
   </span>
   <%= error_tag f, :cdoctor_id %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Patient's GP:</span>
   <%= select f, :gp_id, @gps, prompt: "Choose a GP", class: "form-control" %>
   <span class="input-group-addon">
     <a class="badge badge-warning" drab-click="refresh_gps">Refresh</a>
   </span>
   <%= error_tag f, :gp_id %>
 </div></br>

  <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">PH number:</span>
   <%= text_input f, :ph_number, class: "form-control", style: "display:inline;" %>
   <%= error_tag f, :ph_number %><p></p>
</div></br>
```

```
 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">PPSN:</span>
   <%= text_input f, :ppsn, class: "form-control", style: "display:inline;" %>
   <%= error_tag f, :ppsn %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Gender:</span>
   <%= select f, :gender, @genders, class: "form-control" %>
   <span class="input-group-addon">
     <a class="badge badge-warning" drab-click="refresh_genders">Refresh</a>
   </span>
   <%= error_tag f, :gender %>
 </div></br>
  <div class="input-group">
   <span class="input-group-addon" id="sizing-addon1">Does the patient have a
medical card?</span></span>
   <%= checkbox f, :medical_card_present, class: "checkbox", id: "medical_card",
onclick: "ShowHideMedicalCard(this)", style: "width: 30px; height: 30px;
margin-left: 20px;"  %>
   <%= error_tag f, :medical_card_present %>
 </div></br>

 <div id="medical_card_number"class="alert alert-info" style="display: none">
   <div class="input-group" >
       <span class="input-group-addon" id="basic-addon1">Medical card
number:</span>
       <%= text_input f, :medical_card_number, class: "form-control"  %>
       <%= error_tag f, :medical_card_number %>
   </div></br>

   <div class="input-group" >
       <span class="input-group-addon" id="basic-addon1">Medical card
expiry:</span>
       <%= date_select f, :medical_card_expiry, class: "form-control"%>
       <%= error_tag f, :medical_card_expiry %>
   </div>
```

```
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Mobile number:</span>
   <%= text_input f, :mobile_number, class: "form-control" %>
   <%= error_tag f, :mobile_number %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Landline number:</span>
   <%= text_input f, :landline_number, class: "form-control" %>
   <%= error_tag f, :landline_number %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Is the patient currently
awating transfer?:</span>
     <%= checkbox f, :transfer_status, class: "checkbox ", style: "width: 30px;
height: 30px; margin-left: 20px;" %>
     <%= error_tag f, :transfer_status %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Is the patient currently
active?:</span>
     <%= checkbox f, :active, class: "checkbox ", id: "active", onclick:
"ShowHideActive(this)", style: "width: 30px; height: 30px; margin-left: 20px;" %>
     <%= error_tag f, :active %>
 </div></br>


 <div id="active_details" class="alert alert-info" style="display: block">
   <div class="input-group">
     <span class="input-group-addon" id="basic-addon1">Inactivity details:</span>
     <%= select(f, :active_details, @inactivity, prompt: "Choose an inactivity
reason", class: "form-control" )%>
     <%= error_tag f, :active_details %>
     </div>
```

```
 </div></br>

 </br><h2>Next of kin details</h2>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Relationship to
patient:</span>
   <%= select f, :nok_rel, @relations , class: "form-control" %>
   <span class="input-group-addon">
     <a class="badge badge-warning" drab-click="refresh_relations">Refresh</a>
   </span>
   <%= error_tag f, :nok_rel %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Name:</span>
   <%= text_input f, :nok_name, class: "form-control" %>
   <%= error_tag f, :nok_name %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Address:</span>
   <%= text_input f, :nok_address, class: "form-control" %>
   <%= error_tag f, :nok_address %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Mobile number:</span>
   <%= text_input f, :nok_mobile_number, class: "form-control" %>
   <%= error_tag f, :nok_mobile_number %>
 </div></br>

 <div class="input-group">
   <span class="input-group-addon" id="basic-addon1">Landline number:</span>
   <%= text_input f, :nok_landline_number, class: "form-control" %>
   <%= error_tag f, :nok_landline_number %>
 </div></br>
```

```
 <div class="form-group">
   <%= submit "Save", class: "btn btn-primary btn" %>
 </div>
<% end %>

<script type="text/javascript">
   function ShowHideActive(active) {
       var active_details = document.getElementById("active_details");
       active_details.style.display = active.checked ? "none" : "block";
   }
   function ShowHideMedicalCard(active) {
       var medical_card_number = document.getElementById("medical_card_number");
       var medical_card_expiry = document.getElementById("medical_card_expiry");
       medical_card_number.style.display = active.checked ? "block" : "none";
       medical_card_expiry.style.display = active.checked ? "block" : "none";
   }
</script>
```

Misc. Files

Patient_commander.ex

```
defmodule Cmsv1.PatientCommander do
 use Drab.Commander

 alias Cmsv1.Pharmacy
 alias Cmsv1.Repo
 alias Cmsv1.Patient
 alias Cmsv1.CDoctor
 alias Cmsv1.GP
 alias Cmsv1.Inactivity
 alias Cmsv1.Gender
 alias Cmsv1.Relationship


 # function to refresh gender select input
 def refresh_genders(socket, _sender) do
   genders = Repo.all(Gender) |> Enum.map(&{&1.gender, &1.gender}) |>
Enum.into(%{})
   set_prop socket, "#patient_gender" , options: genders
 end

 # function to refresh pharmacy select input
 def refresh_pharms(socket, _sender) do
   pharms = Repo.all(Pharmacy) |> Enum.map(&{&1.pharm_id, &1.name<>" ,
"<>&1.address}) |> Enum.into(%{})
   set_prop socket, "#patient_pharm_id" , options: pharms
 end

 # function to refresh general practitioner select input
 def refresh_gps(socket, _sender) do
   gps = Repo.all(GP) |> Enum.map(&{&1.gp_id, &1.name}) |> Enum.into(%{})
```

```
    set_prop socket, "#patient_gp_id" , options: gps
  end

  # function to refresh doctor select input
  def refresh_doctors(socket, _sender) do
    doctors = Repo.all(CDoctor) |> Enum.map(&{&1.cdoctor_id,&1.name}) |>
Enum.into(%{})
    set_prop socket, "#patient_cdoctor_id" , options: doctors
  end

  # function to refresh relationship select input
  def refresh_relations(socket, _sender) do
    relations = Repo.all(Relationship) |> Enum.map(&{&1.relationship,
&1.relationship}) |> Enum.into(%{})
    set_prop socket, "#patient_nok_rel" , options: relations
  end
end
```

Router.ex

```
defmodule Cmsv1.Router do
 use Cmsv1.Web, :router

 pipeline :browser do
   plug :accepts, ["html"]
   plug :fetch_session
   plug :fetch_flash
   plug :protect_from_forgery
   plug :put_secure_browser_headers
   plug Cmsv1.Auth, repo: Rumbl.Repo
 end

 pipeline :api do
   plug :accepts, ["json"]
 end

 scope "/", Cmsv1 do
   pipe_through :browser # Use the default browser stack
   get "/", PatientController, :index
   get "/patients", PatientController, :index
   get "/Reports", ReportController, :index
   resources "/pharms", PharmacyController
   resources "/gps", GPController
   resources "/cdoctors", CDoctorController
   resources "/patients", PatientController
   resources "/reasons", InactivityController
   resources "/genders", GenderController
   resources "/relationships", RelationshipController
   resources "/phleb", PhlebotomyController
   resources "/vaccs", VaccinationsController
   resources "/vaccbrands", VaccBrandController
   resources "/clinics", ClinicController
   resources "/users", UserController, only: [:index, :show, :new, :create]
```

```
    resources "/sessions", SessionController, only: [:new, :create, :delete]

  end

  # Other scopes may use custom stacks.
  # scope "/api", Cmsv1 do
  #   pipe_through :api
  # end
end
```

Seeds.exs

```
#Reasons
reason = %Cmsv1.Inactivity{reason: "Absent without reason"}
Cmsv1.Repo.insert(reason)
reason = %Cmsv1.Inactivity{reason: "Deceased"}
Cmsv1.Repo.insert(reason)
reason = %Cmsv1.Inactivity{reason: "Prison"}
Cmsv1.Repo.insert(reason)
reason = %Cmsv1.Inactivity{reason: "Unknown"}
Cmsv1.Repo.insert(reason)

# Genders
gender = %Cmsv1.Gender{gender: "Male"}
Cmsv1.Repo.insert(gender)
gender = %Cmsv1.Gender{gender: "Female"}
Cmsv1.Repo.insert(gender)

#Patient Relationships
relation = %Cmsv1.Relationship{relationship: "Brother"}
Cmsv1.Repo.insert(relation)
relation = %Cmsv1.Relationship{relationship: "Sister"}
Cmsv1.Repo.insert(relation)
```

```
relation = %Cmsv1.Relationship{relationship: "Mother"}
Cmsv1.Repo.insert(relation)
relation = %Cmsv1.Relationship{relationship: "Father"}
Cmsv1.Repo.insert(relation)

# Vaccination Brand
vbrand = %Cmsv1.VaccBrand{vaccbrand: "Twinrix"}
Cmsv1.Repo.insert(vbrand)
vbrand = %Cmsv1.VaccBrand{vaccbrand: "HBVAXPRO"}
Cmsv1.Repo.insert(vbrand)

#Clinic
clinic = %Cmsv1.Clinic{name: "defaultclinic", address: "defaultclinic", clinic_id:
"8d0b73bc-2fd4-4b55-b519-361fa367cffb"}
Cmsv1.Repo.insert(clinic)

clinic = %Cmsv1.Clinic{name: "Carlow", address: "St. Dymphnas", clinic_id:
"8a1543c2-6b60-4cda-b4b3-ada49743eb6f"}
Cmsv1.Repo.insert(clinic)

# Users
user = %Cmsv1.User{name: "admin", username: "admin", password_hash:
"$2b$12$IF4JpsOq3bBg0MNZyohSFu0DkBsJMa.eNcnoLU2NeCuBmtXuFbTP2", level: "admin",
clinic_id: "8d0b73bc-2fd4-4b55-b519-361fa367cffb"}
Cmsv1.Repo.insert(user)

user = %Cmsv1.User{name: "Deirdre", username: "deirdre", password_hash:
"$2b$12$IF4JpsOq3bBg0MNZyohSFu0DkBsJMa.eNcnoLU2NeCuBmtXuFbTP2", level: "standard",
clinic_id: "8a1543c2-6b60-4cda-b4b3-ada49743eb6f"}
Cmsv1.Repo.insert(user)

# GPs

gp = %Cmsv1.GP{gp_id: "0d56421d-5728-410b-b091-372c4b2e7801", name: "Joe Bloggs",
address: "Waterford", mobile_number: "085-xxx-xxxx", landline_number:
"05991-xx-xxx", fax: "xxx1234", email: "joe@bloggs.com", training_level: "1"}
Cmsv1.Repo.insert(gp)
```

```
gp = %Cmsv1.GP{gp_id: "0d564111-5728-410b-b091-372c4b2e7801", name: "John Doe",
address: "Carlow", mobile_number: "085-xxx-xxxx", landline_number: "05991-xx-xxx",
fax: "xxx1234", email: "john@doe.com", training_level: "1"}
Cmsv1.Repo.insert(gp)

gp = %Cmsv1.GP{gp_id: "0d564222-5728-410b-b091-372c4b2e7801", name: "Joe Dolan",
address: "Baltinglass", mobile_number: "085-xxx-xxxx", landline_number:
"05991-xx-xxx", fax: "xxx1234", email: "joe@dolan.com", training_level: "1"}
Cmsv1.Repo.insert(gp)

gp = %Cmsv1.GP{gp_id: "0d564333-5728-410b-b091-372c4b2e7801", name: "Donal Drump",
address: "Kilkenny", mobile_number: "085-xxx-xxxx", landline_number:
"05991-xx-xxx", fax: "xxx1234", email: "donal@drump.com", training_level: "1"}
Cmsv1.Repo.insert(gp)

# Clinic Doctors

cdoctor = %Cmsv1.CDoctor{cdoctor_id: "1a05503a-d40d-42f7-84df-c3fe3fcd531e", name:
"Jim Dunne", address: "Baltinglass", mobile_number: "085-xxx-xxxx",
landline_number: "05991-xx-xxx", fax: "xxx1234", email: "jim@bloggs.com",
training_level: "2"}
Cmsv1.Repo.insert(cdoctor)

cdoctor = %Cmsv1.CDoctor{cdoctor_id: "1a055111-d40d-42f7-84df-c3fe3fcd531e", name:
"Declan Byrne", address: "Carlow", mobile_number: "085-xxx-xxxx", landline_number:
"05991-xx-xxx", fax: "xxx1234", email: "declan@byrne.com", training_level: "2"}
Cmsv1.Repo.insert(cdoctor)

cdoctor = %Cmsv1.CDoctor{cdoctor_id: "1a055222-d40d-42f7-84df-c3fe3fcd531e", name:
"Jimmy McNamara", address: "Kilkenny", mobile_number: "085-xxx-xxxx",
landline_number: "05991-xx-xxx", fax: "xxx1234", email: "caroline@nolan.com",
training_level: "2"}
Cmsv1.Repo.insert(cdoctor)

cdoctor = %Cmsv1.CDoctor{cdoctor_id: "1a055333-d40d-42f7-84df-c3fe3fcd531e", name:
"Caroline Nolan", address: "Waterford", mobile_number: "085-xxx-xxxx",
landline_number: "05991-xx-xxx", fax: "xxx1234", email: "jimmy@mcnamara.com",
training_level: "2"}
```

```
Cmsv1.Repo.insert(cdoctor)

#Pharmacies

p = %Cmsv1.Pharmacy{pharm_id: "3480f2c1-ccf3-46bb-93c2-e5c1ac8d8208", name: "Lloyds
Carlow", address: "Main Street, Carlow Town, Co. Carlow", mobile_number:
"085-xxx-xxxx", landline_number: "05991-xx-xxx", fax: "xxx1234", email:
"Morriseys@carlow.com", contact_person: "Jenny Dooley", contact_person_number:
"083-xxx-xxxx"}
Cmsv1.Repo.insert(p)

p = %Cmsv1.Pharmacy{pharm_id: "3480f111-ccf3-46bb-93c2-e5c1ac8d8208", name:
"Morrisseys Carlow", address: "Tullow Street, Carlow Town, Co. Carlow",
mobile_number: "085-xxx-xxxx", landline_number: "05991-xx-xxx", fax: "xxx1234",
email: "lloyds@carlow.com", contact_person: "Jenny Byrne", contact_person_number:
"087-xxx-xxxx"}
Cmsv1.Repo.insert(p)

p = %Cmsv1.Pharmacy{pharm_id: "3480f222-ccf3-46bb-93c2-e5c1ac8d8208", name: "Lloyds
Carlow", address: "Barrow Street, Carlow Town, Co. Carlow", mobile_number:
"085-xxx-xxxx", landline_number: "05991-xx-xxx", fax: "xxx1234", email:
"Morriseys@carlow.com", contact_person: "Jenny Dooley", contact_person_number:
"083-xxx-xxxx"}
Cmsv1.Repo.insert(p)

p = %Cmsv1.Pharmacy{pharm_id: "3480f333-ccf3-46bb-93c2-e5c1ac8d8208", name: "Sam
McCauleys Carlow", address: "Other Street, Carlow Town, Co. Carlow", mobile_number:
"085-xxx-xxxx", landline_number: "05991-xx-xxx", fax: "xxx1234", email:
"lloyds@carlow.com", contact_person: "Jenny Byrne", contact_person_number:
"087-xxx-xxxx"}
Cmsv1.Repo.insert(p)

p = %Cmsv1.Pharmacy{pharm_id: "3480f444-ccf3-46bb-93c2-e5c1ac8d8208", name: "Sam
McCauleys Waterford", address: "Main Street, Waterford Town, Co. Waterford",
mobile_number: "085-xxx-xxxx", landline_number: "05991-xx-xxx", fax: "xxx1234",
email: "Morriseys@carlow.com", contact_person: "Jenny Dooley",
contact_person_number: "083-xxx-xxxx"}
Cmsv1.Repo.insert(p)
```

Mix.exs

```
defmodule Cmsv1.Mixfile do
 use Mix.Project

 def project do
   [app: :cmsv1,
    version: "0.0.1",
    elixir: "~> 1.2",
    elixirc_paths: elixirc_paths(Mix.env),
    compilers: [:phoenix, :gettext] ++ Mix.compilers,
    build_embedded: Mix.env == :prod,
    start_permanent: Mix.env == :prod,
    aliases: aliases(),
    deps: deps()]
 end

 def application do
   [mod: {Cmsv1, []},
    applications: [:phoenix, :phoenix_pubsub, :phoenix_html, :cowboy, :logger,
:gettext,
                   :phoenix_ecto, :postgrex, :drab, :comeonin, :cloak]]
 end

 # Specifies which paths to compile per environment.
 defp elixirc_paths(:test), do: ["lib", "web", "test/support"]
 defp elixirc_paths(_),     do: ["lib", "web"]

 # Specifies your project dependencies.
 #
 # Type `mix help deps` for examples and options.
```

```
 defp deps do
   [{:phoenix, "~> 1.2.5"},
    {:phoenix_pubsub, "~> 1.0"},
    {:phoenix_ecto, "~> 3.0"},
    {:postgrex, ">= 0.0.0"},
    {:phoenix_html, "~> 2.6"},
    {:phoenix_live_reload, "~> 1.0", only: :dev},
    {:gettext, "~> 0.11"},
    {:cowboy, "~> 1.0"},
    {:calendar, "~> 0.17.2"},
    {:drab, "~> 0.7"},
    {:comeonin, "~> 2.0"},
    {:cloak, "~> 0.6.2"}
   ]
 end

 # Aliases are shortcuts or tasks specific to the current project.
 # For example, to create, migrate and run the seeds file at once:
 #
 #     $ mix ecto.setup
 #
 # See the documentation for `Mix` for more info on aliases.
 defp aliases do
   ["ecto.setup": ["ecto.create", "ecto.migrate", "run priv/repo/seeds.exs"],
    "ecto.reset": ["ecto.drop", "ecto.setup"],
    "test": ["ecto.create --quiet", "ecto.migrate", "test"]]
 end
end
```

Web.ex

```
defmodule Cmsv1.Web do
  def model do
   quote do
     use Ecto.Schema

     import Ecto
     import Ecto.Changeset
     import Ecto.Query
   end
 end

 def controller do
   quote do
     use Phoenix.Controller

     alias Cmsv1.Repo
     import Ecto
     import Ecto.Query

     import Cmsv1.Router.Helpers
     import Cmsv1.Gettext
   end
 end

 def view do
   quote do
     use Phoenix.View, root: "web/templates"

     # Import convenience functions from controllers
     import Phoenix.Controller, only: [get_csrf_token: 0, get_flash: 2,
view_module: 1]

     # Use all HTML functionality (forms, tags, etc)
```

```
      use Phoenix.HTML

      import Cmsv1.Router.Helpers
      import Cmsv1.ErrorHelpers
      import Cmsv1.Gettext
    end
  end

  def router do
    quote do
      use Phoenix.Router
    end
  end

  def channel do
    quote do
      use Phoenix.Channel

      alias Cmsv1.Repo
      import Ecto
      import Ecto.Query
      import Cmsv1.Gettext
    end
  end

  defmacro __using__(which) when is_atom(which) do
    apply(__MODULE__, which, [])
  end
end
```

Config.exs

```
use Mix.Config

# General application configuration
config :cmsv1,
 ecto_repos: [Cmsv1.Repo]

# Configures the endpoint
config :cmsv1, Cmsv1.Endpoint,
 url: [host: "localhost"],
 secret_key_base:
"j9VrLxZpJO8fO9A5yaApqaqTyBVG8VHF+Bo8moPbyVgSlRirqJwDpAvG86j++yH1",
 render_errors: [view: Cmsv1.ErrorView, accepts: ~w(html json)],
 pubsub: [name: Cmsv1.PubSub,
          adapter: Phoenix.PubSub.PG2]

# Configures Elixir's Logger
config :logger, :console,
 format: "$time $metadata[$level] $message\n",
 metadata: [:request_id]

config :phoenix, :template_engines,
 drab: Drab.Live.Engine

config :cloak, Cloak.AES.CTR,
 tag: "AES",
 default: true,
 keys: [
   %{tag: <<1>>, key:
:base64.decode("6+o9eXdw90rsUv8tlQSjm9Alq1OXXA8kaB9PvZeMRoQ="), default: true}
 ]

# Import environment specific config. This must remain at the bottom
```

```
# of this file so it overrides the configuration defined above.
import_config "#{Mix.env}.exs"
```

User_socket.ex

```
defmodule Cmsv1.UserSocket do
 use Phoenix.Socket
 use Drab.Socket

  transport :websocket, Phoenix.Transports.WebSocket
  def connect(_params, socket) do
   {:ok, socket}
 end

  def id(_socket), do: nil
end
```