

# Choices! Counseling Services Database

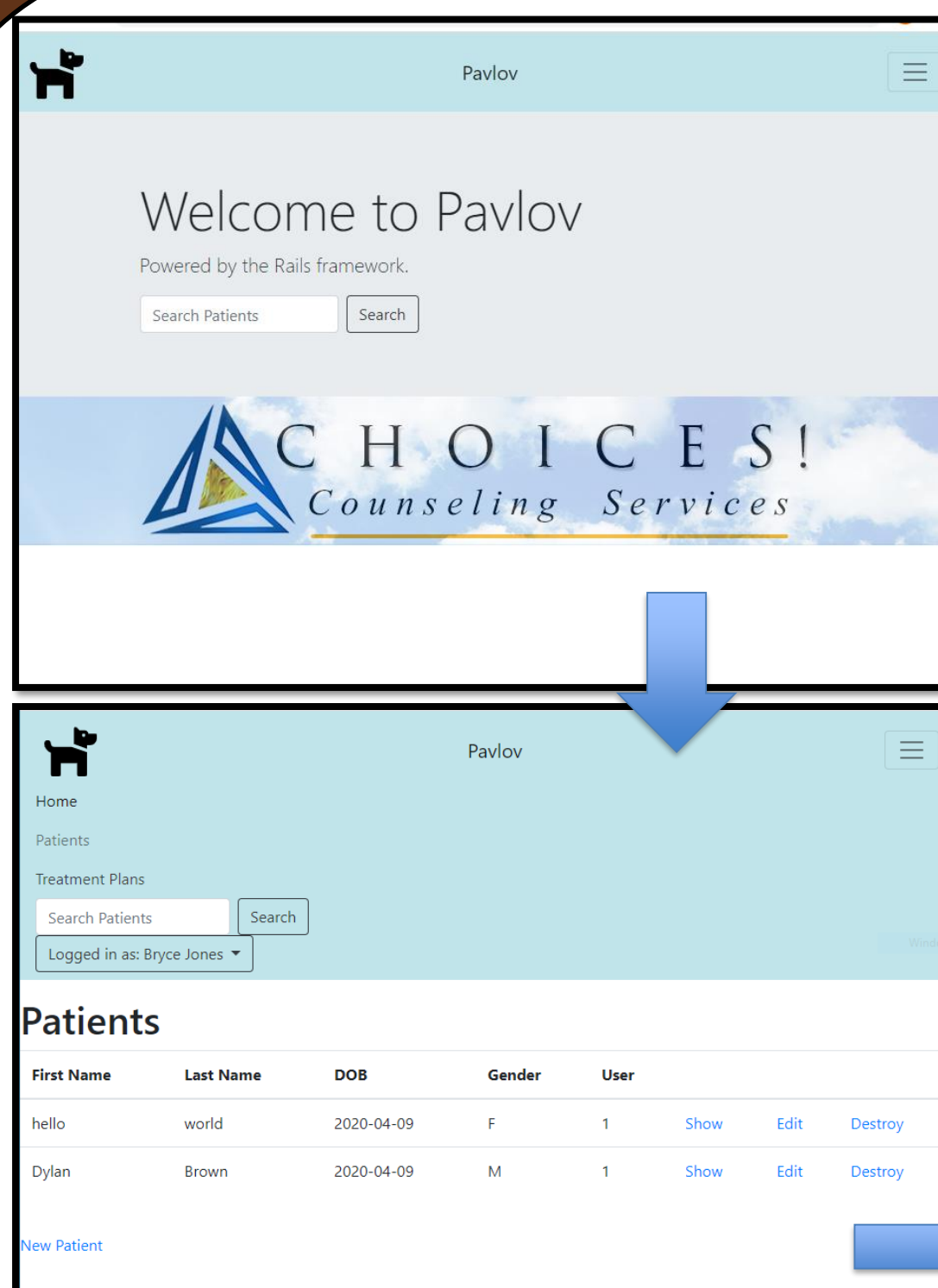
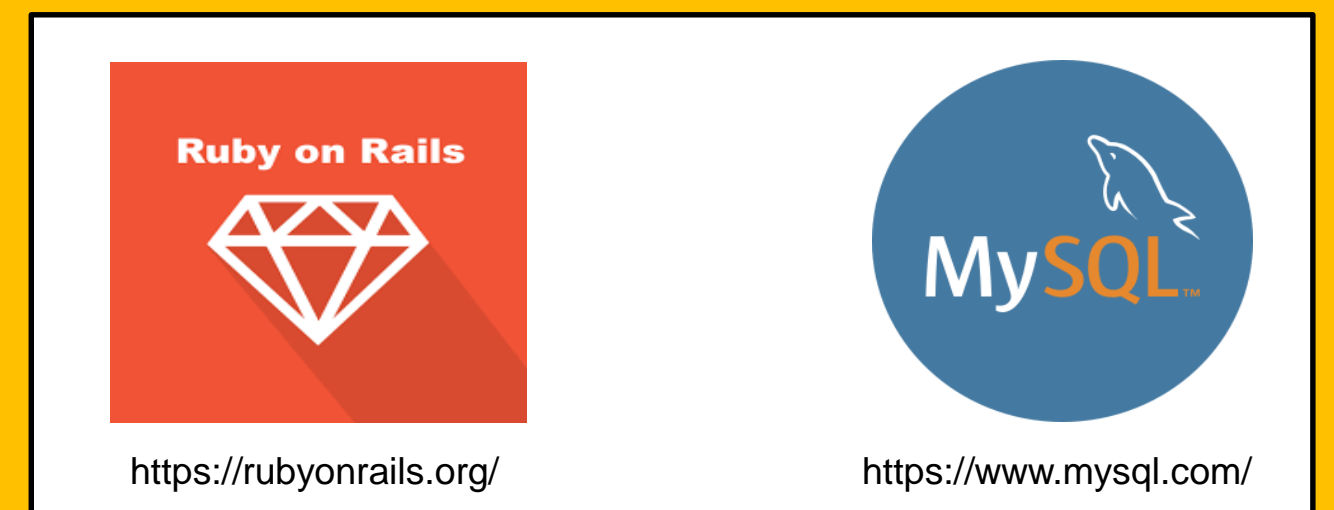
Pavlov Team – Kenyona Parker – Dylan Brown – Abby Palmer – Bryce Jones – Deven Harris

## Abstract

- This project is implementing and deploying a functional patient database with customizable treatment plans and progress notes for Choices!, a counseling service center.
- To build this application and support the various requirements, several frameworks were considered for this project including Django, Flask, and Ruby on Rails.
- Existing libraries within Rails allow for a choice of several database engines; MySQL, Postgresql, MongoDB, and NoSQL were considered. MySQL was adopted as a result of the extensive documentation and for continuity with the previous solution. The Ruby on Rails suite was selected for reasons of existing familiarity for the team.
- Front-end requirements were met with a combination of HTML and Javascript, linked to a Bootstrap 5 framework in order to streamline user accessibility.
- A key quality desired by the customer was ease-of-use, to encourage higher efficiency and easy adoption by the stakeholders.
- In the process of development, secure data handling also emerged as a further and highly-desired attribute.

## Platforms and Tools

- Ruby on Rails (with MySQL database)
- JavaScript
- HTML 5
- Bootstrap 5



```
class PatientsController < ApplicationController
  before_action :set_patient, only: [:show, :edit, :update, :destroy]

  # GET /patients
  # GET /patients.json
  def index
    if params[:search]
      @patients = Patient.search_by_full_name(params[:search])
    else
      @patients = Patient.all
    end
  end

  # GET /patients/1
  # GET /patients/1.json
  def show
  end

  # GET /patients/new
  def new
    @patient = Patient.new
  end

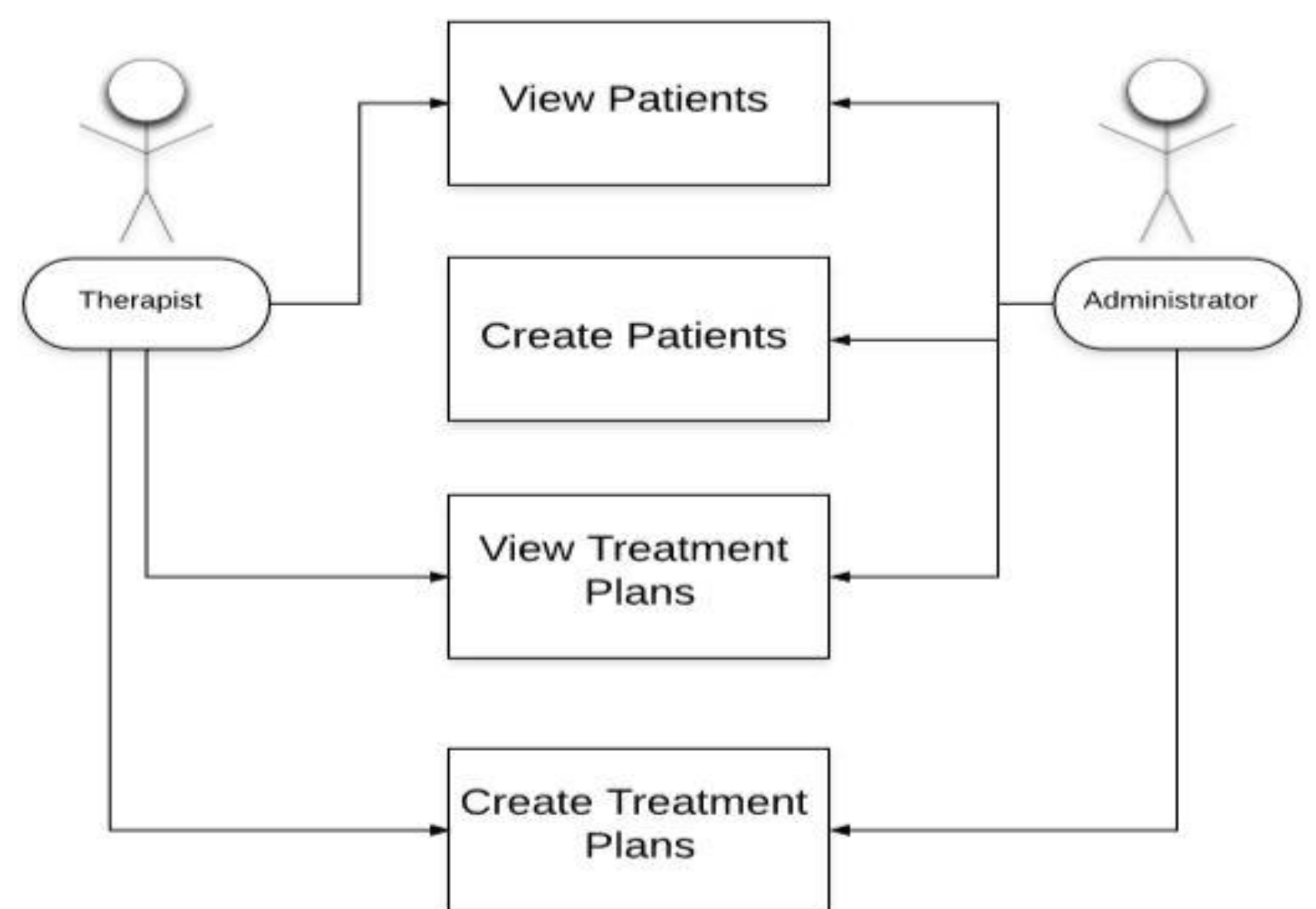
  # GET /patients/1/edit
  def edit
  end

  # POST /patients
  # POST /patients.json
  def create
    @patient = Patient.new(patient_params)

    respond_to do |format|
      if @patient.save
        format.html { redirect_to @patient, notice: "Patient was successfully created." }
        format.json { render :show, status: :created, location: @patient }
      else
        format.html { render :new }
        format.json { render :json: @patient.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /patients/1
  # PATCH/PUT /patients/1.json
  def update
    respond_to do |format|
      if @patient.update(patient_params)
        format.html { redirect_to @patient, notice: "Patient was successfully updated." }
        format.json { render :show, status: :ok, location: @patient }
      else
        format.html { render :edit }
        format.json { render :json: @patient.errors, status: :unprocessable_entity }
      end
    end
  end
end
```

## USE CASE DIAGRAM



## Additional Features

- Created google forms that are able to be uploaded for treatment plans
- Login in screen with password

## Challenges

- System was built to be a potential successor to a currently deployed solution, but source code and design/specifications were unavailable
- Circumstances including the COVID-19 situation required rapid changes to team practices due to dispersal of team and move to network-based communications
- Some features required significantly more complexity and customization capability than originally anticipated; individual treatment plan templates in particular required architectural changes

## Future Work

- Added ability to create progress notes based on information previously recorded in the creation of patients and treatment plans
- Implementing the real data from the counseling database into the system to be used
- Authorizing access to real users in the counseling staff
- Added security in accessing data for both viewing and manipulating for patient confidentiality

## Acknowledgements

- Professor Paula Dranger, MSW, LCSW, MAC
- Professor Nick Rosasco, DSc



VALPARAISO  
UNIVERSITY

Computing and  
Information Sciences