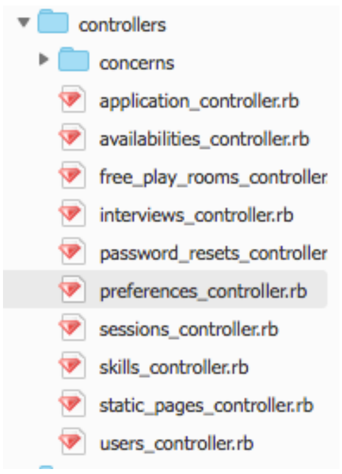
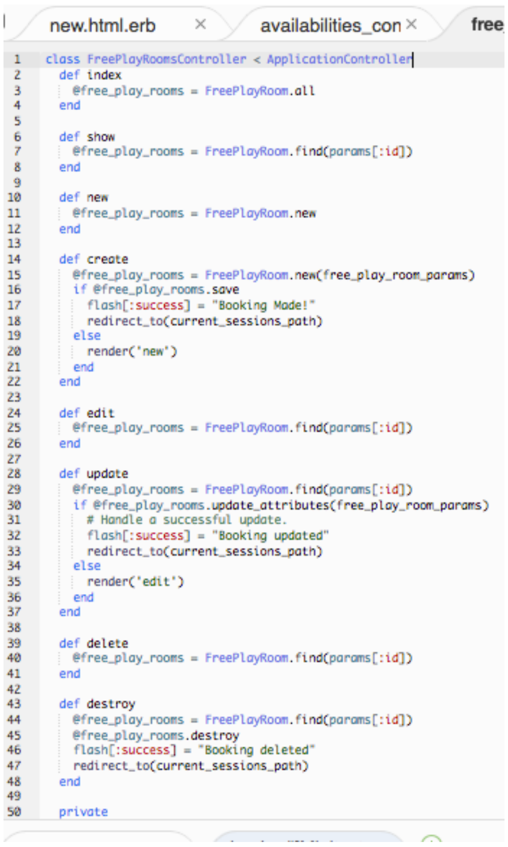
Controllers:



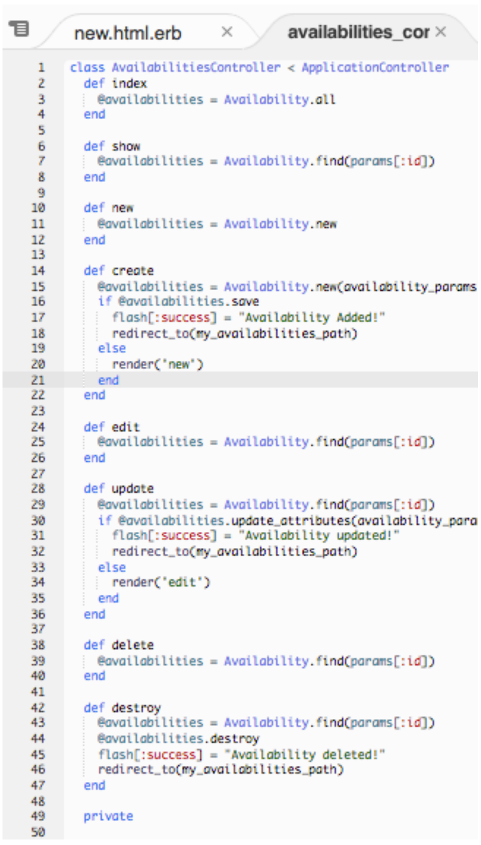
```
controllers
  concerns
  application_controller.rb
  availabilities_controller.rb
  free_play_rooms_controller
  interviews_controller.rb
  password_resets_controller
  preferences_controller.rb
  sessions_controller.rb
  skills_controller.rb
  static_pages_controller.rb
  users_controller.rb
```

Figure 1. All controllers as of the 16th October



```ruby
class FreePlayRoomsController < ApplicationController
  def index
    @free_play_rooms = FreePlayRoom.all
  end

  def show
    @free_play_rooms = FreePlayRoom.find(params[:id])
  end

  def new
    @free_play_rooms = FreePlayRoom.new
  end

  def create
    @free_play_rooms = FreePlayRoom.new(free_play_room_params)
    if @free_play_rooms.save
      flash[:success] = "Booking Made!"
      redirect_to(current_sessions_path)
    else
      render('new')
    end
  end

  def edit
    @free_play_rooms = FreePlayRoom.find(params[:id])
  end

  def update
    @free_play_rooms = FreePlayRoom.find(params[:id])
    if @free_play_rooms.update_attributes(free_play_room_params)
      # Handle a successful update.
      flash[:success] = "Booking updated"
      redirect_to(current_sessions_path)
    else
      render('edit')
    end
  end

  def delete
    @free_play_rooms = FreePlayRoom.find(params[:id])
  end

  def destroy
    @free_play_rooms = FreePlayRoom.find(params[:id])
    @free_play_rooms.destroy
    flash[:success] = "Booking deleted"
    redirect_to(current_sessions_path)
  end

  private
```

Figure 2. Free play Room Controller

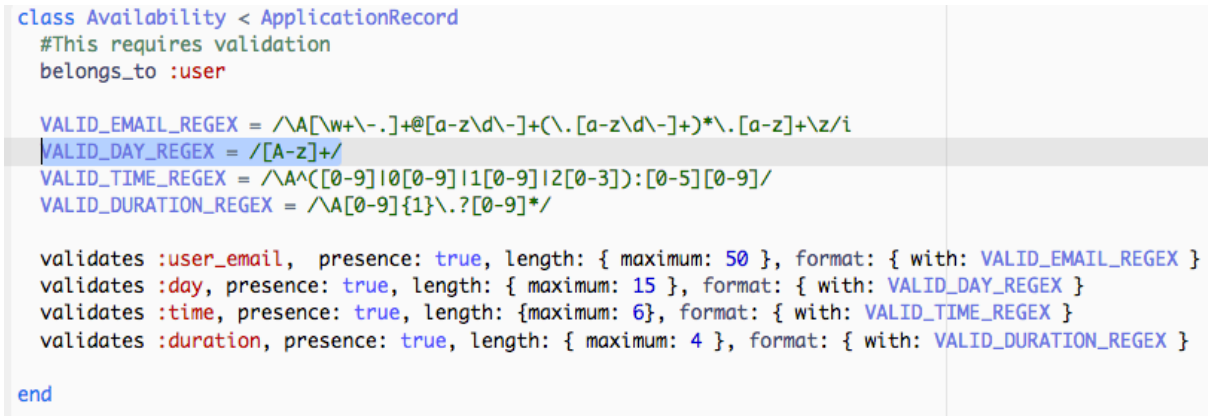


```ruby
class AvailabilitiesController < ApplicationController
  def index
    @availabilities = Availability.all
  end

  def show
    @availabilities = Availability.find(params[:id])
  end

  def new
    @availabilities = Availability.new
  end

  def create
    @availabilities = Availability.new(availability_params
    if @availabilities.save
      flash[:success] = "Availability Added!"
      redirect_to(my_availabilities_path)
    else
      render('new')
    end
  end

  def edit
    @availabilities = Availability.find(params[:id])
  end

  def update
    @availabilities = Availability.find(params[:id])
    if @availabilities.update_attributes(availability_para
      flash[:success] = "Availability updated!"
      redirect_to(my_availabilities_path)
    else
      render('edit')
    end
  end

  def delete
    @availabilities = Availability.find(params[:id])
  end

  def destroy
    @availabilities = Availability.find(params[:id])
    @availabilities.destroy
    flash[:success] = "Availability deleted!"
    redirect_to(my_availabilities_path)
  end

  private

```

Figure 3. Availabilities Controller

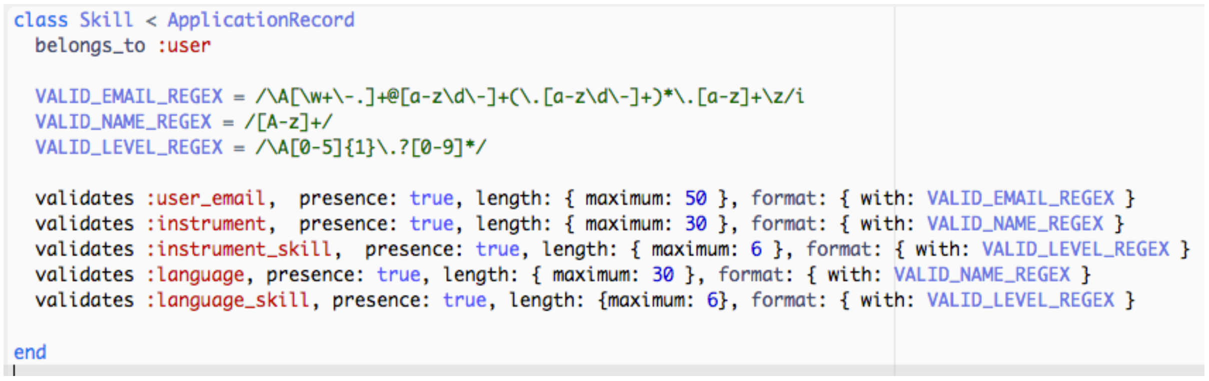
Controllers where implemented for all modules stated. As can be seen in Figure 1, when generated via rails, the script for the controllers where automatically generated in the controller's folder with the names given. For all the modules I added, there controllers remained basically the same with some minor name changes and object changes. The only real noticeable distinction between the controllers is the mass permit assignment parameters. The parameters where changed to correspond with their respective table.

Models:



```ruby
class Availability < ApplicationRecord
  #This requires validation
  belongs_to :user

  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z\d\-]+)*\.[a-z]+\z/i
  VALID_DAY_REGEX = /[A-z]+/
  VALID_TIME_REGEX = /\A^([0-9]|0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]/
  VALID_DURATION_REGEX = /\A[0-9]{1}\.?[0-9]*/

  validates :user_email,  presence: true, length: { maximum: 50 }, format: { with: VALID_EMAIL_REGEX }
  validates :day, presence: true, length: { maximum: 15 }, format: { with: VALID_DAY_REGEX }
  validates :time, presence: true, length: {maximum: 6}, format: { with: VALID_TIME_REGEX }
  validates :duration, presence: true, length: { maximum: 4 }, format: { with: VALID_DURATION_REGEX }

end
```

Figure 4. Availability Model

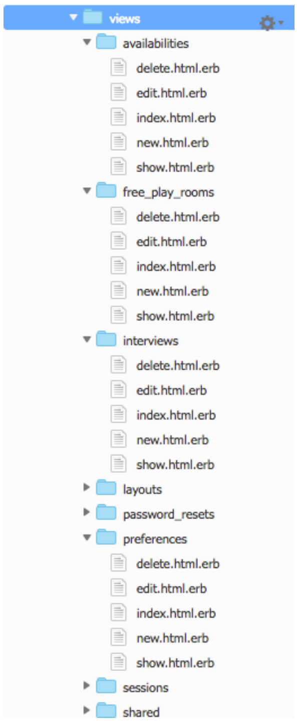


```ruby
class Skill < ApplicationRecord
  belongs_to :user

  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z\d\-]+)*\.[a-z]+\z/i
  VALID_NAME_REGEX = /[A-z]+/
  VALID_LEVEL_REGEX = /\A[0-5]{1}\.?[0-9]*/

  validates :user_email,  presence: true, length: { maximum: 50 }, format: { with: VALID_EMAIL_REGEX }
  validates :instrument,  presence: true, length: { maximum: 30 }, format: { with: VALID_NAME_REGEX }
  validates :instrument_skill,  presence: true, length: { maximum: 6 }, format: { with: VALID_LEVEL_REGEX }
  validates :language, presence: true, length: { maximum: 30 }, format: { with: VALID_NAME_REGEX }
  validates :language_skill, presence: true, length: {maximum: 6}, format: { with: VALID_LEVEL_REGEX }

end
```

Figure 5. Skill Model

Just like the controllers, most of the models for all modules remains relatively the same with the exception of small changes depending on the data types. The figures above (Figure 4 and 5) show the similarities between two models. Additionally, they show the validation for each input. In all modules, each input is required and in most cases has a regular expression that it must match.

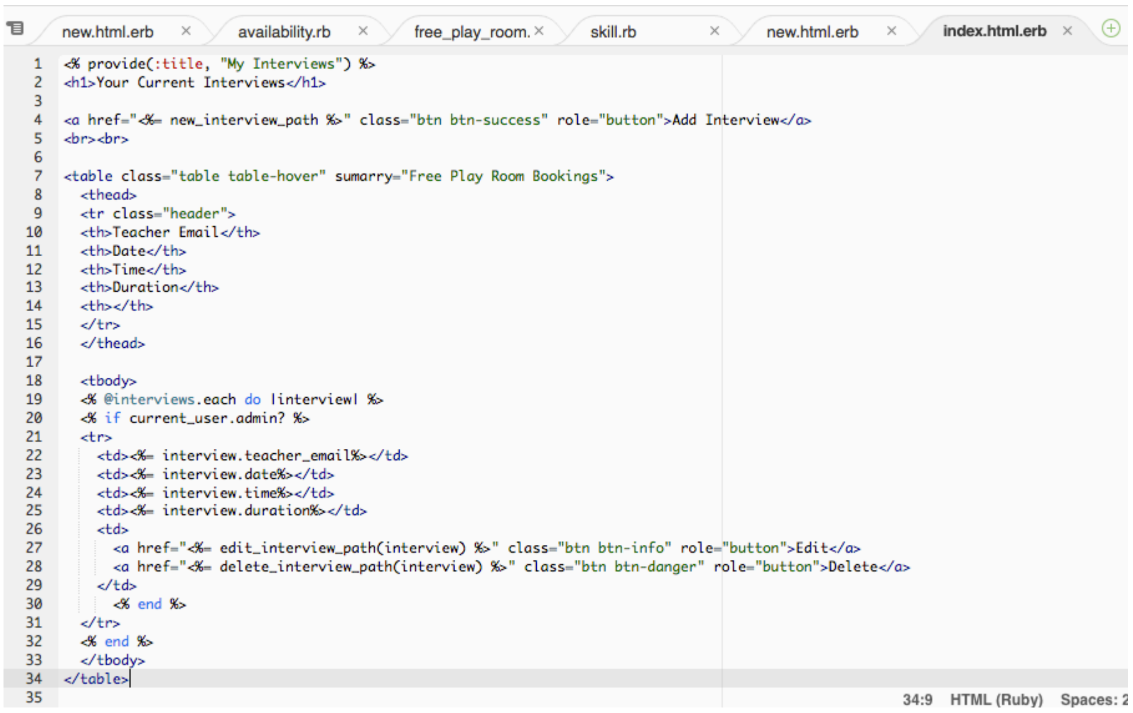
Views:



```
▾ views
  ▾ availabilities
      delete.html.erb
      edit.html.erb
      index.html.erb
      new.html.erb
      show.html.erb
  ▾ free_play_rooms
      delete.html.erb
      edit.html.erb
      index.html.erb
      new.html.erb
      show.html.erb
  ▾ interviews
      delete.html.erb
      edit.html.erb
      index.html.erb
      new.html.erb
      show.html.erb
  ▸ layouts
  ▸ password_resets
  ▾ preferences
      delete.html.erb
      edit.html.erb
      index.html.erb
      new.html.erb
      show.html.erb
  ▸ sessions
  ▸ shared
```

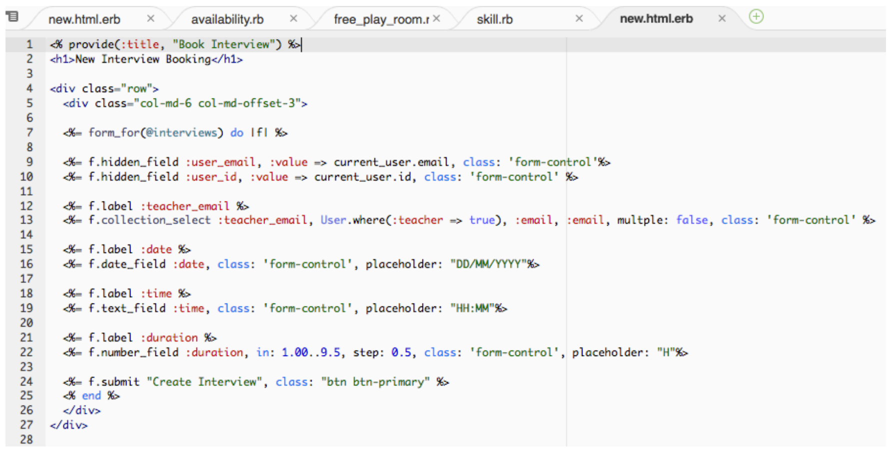
Each controller has a page index, show, new, edit and delete. This can be seen in Figure 6. These pages allow the users to manipulate the database. This is user side of the CRUD operation. All of these pages have been edited and either display the forms required or index the current data in a table. The only page unused is the show page. At this current stage in development it has been decided to not use the show page however generate it encase needed for future tasks.

Figure 6. Views for the Modules.



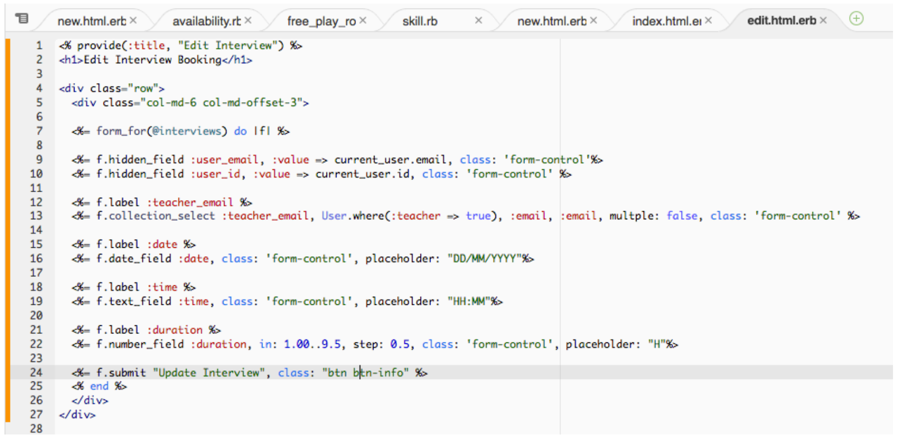
```erb
<% provide(:title, "My Interviews") %>
<h1>Your Current Interviews</h1>

<a href="<%= new_interview_path %>" class="btn btn-success" role="button">Add Interview</a>
<br><br>

<table class="table table-hover" sumarry="Free Play Room Bookings">
  <thead>
  <tr class="header">
  <th>Teacher Email</th>
  <th>Date</th>
  <th>Time</th>
  <th>Duration</th>
  <th></th>
  </tr>
  </thead>

  <tbody>
  <% @interviews.each do |interview| %>
  <% if current_user.admin? %>
  <tr>
    <td><%= interview.teacher_email%></td>
    <td><%= interview.date%></td>
    <td><%= interview.time%></td>
    <td><%= interview.duration%></td>
    <td>
      <a href="<%= edit_interview_path(interview) %>" class="btn btn-info" role="button">Edit</a>
      <a href="<%= delete_interview_path(interview) %>" class="btn btn-danger" role="button">Delete</a>
    </td>
      <% end %>
  </tr>
  <% end %>
  </tbody>
</table>
```

Figure 7. Index Page for Interviews



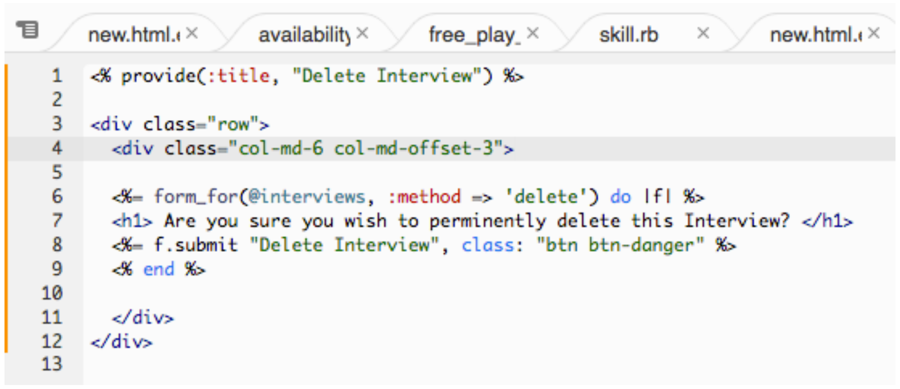
```erb
<% provide(:title, "Book Interview") %>
<h1>New Interview Booking</h1>

<div class="row">
  <div class="col-md-6 col-md-offset-3">

  <%= form_for(@interviews) do |f| %>

  <%= f.hidden_field :user_email, :value => current_user.email, class: 'form-control'%>
  <%= f.hidden_field :user_id, :value => current_user.id, class: 'form-control' %>

  <%= f.label :teacher_email %>
  <%= f.collection_select :teacher_email, User.where(:teacher => true), :email, :email, multple: false, class: 'form-control' %>

  <%= f.label :date %>
  <%= f.date_field :date, class: 'form-control', placeholder: "DD/MM/YYYY"%>

  <%= f.label :time %>
  <%= f.text_field :time, class: 'form-control', placeholder: "HH:MM"%>

  <%= f.label :duration %>
  <%= f.number_field :duration, in: 1.00..9.5, step: 0.5, class: 'form-control', placeholder: "H"%>

  <%= f.submit "Create Interview", class: "btn btn-primary" %>
  <% end %>
  </div>
</div>
```

Figure 8. New Page for Interviews



```erb
<% provide(:title, "Edit Interview") %>
<h1>Edit Interview Booking</h1>

<div class="row">
  <div class="col-md-6 col-md-offset-3">

  <%= form_for(@interviews) do |f| %>

  <%= f.hidden_field :user_email, :value => current_user.email, class: 'form-control'%>
  <%= f.hidden_field :user_id, :value => current_user.id, class: 'form-control' %>

  <%= f.label :teacher_email %>
  <%= f.collection_select :teacher_email, User.where(:teacher => true), :email, :email, multple: false, class: 'form-control' %>

  <%= f.label :date %>
  <%= f.date_field :date, class: 'form-control', placeholder: "DD/MM/YYYY"%>

  <%= f.label :time %>
  <%= f.text_field :time, class: 'form-control', placeholder: "HH:MM"%>

  <%= f.label :duration %>
  <%= f.number_field :duration, in: 1.00..9.5, step: 0.5, class: 'form-control', placeholder: "H"%>

  <%= f.submit "Update Interview", class: "btn btn-info" %>
  <% end %>
  </div>
</div>
```

Figure 9. Edit Page for Interviews

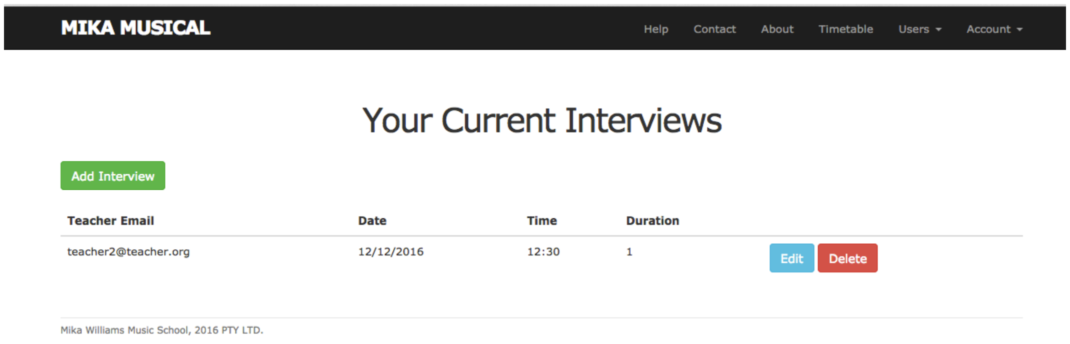


```erb
<% provide(:title, "Delete Interview") %>

<div class="row">
  <div class="col-md-6 col-md-offset-3">

  <%= form_for(@interviews, :method => 'delete') do |f| %>
  <h1> Are you sure you wish to perminently delete this Interview? </h1>
  <%= f.submit "Delete Interview", class: "btn btn-danger" %>
  <% end %>

  </div>
</div>
```

Figure 10. Delete Page for Interviews

The index, new, edit and delete pages for all modules are structured identically. Index displays the data for that particular user in a table. Each element in that table can either be deleted or edited. Where applicable, select drop down boxes were used to make adding information easier for the user. Each for completes the intended operation.

Final Module View:

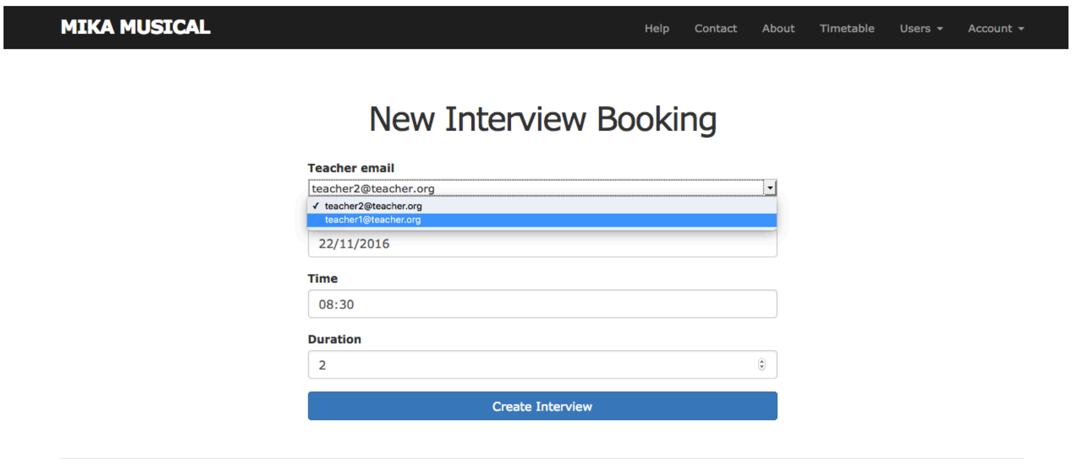


Your Current Interviews

Teacher Email	Date	Time	Duration	
teacher2@teacher.org	12/12/2016	12:30	1	Edit Delete

Mika Williams Music School, 2016 PTY LTD.

Figure 11. Index View for Interviews



New Interview Booking

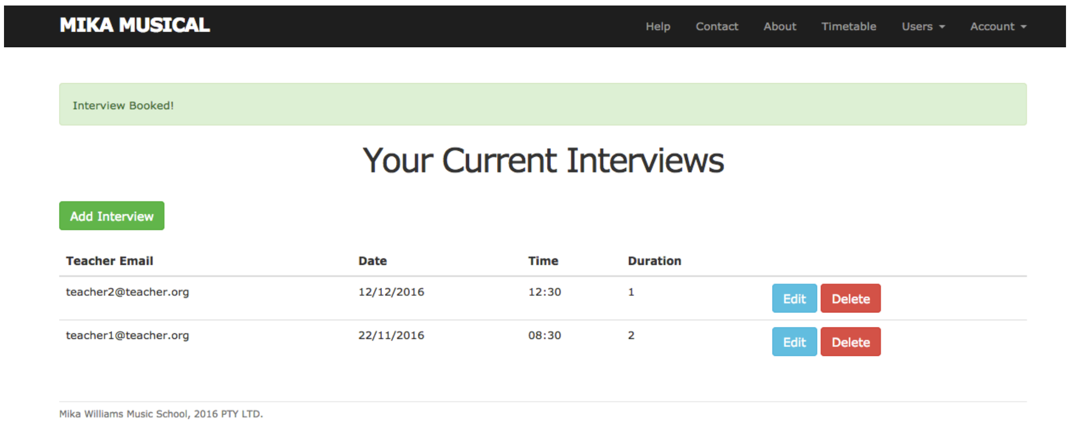
Teacher email: teacher2@teacher.org
✓ teacher2@teacher.org
teacher1@teacher.org

22/11/2016

Time: 08:30

Duration: 2

Figure 12. New View for Interviews



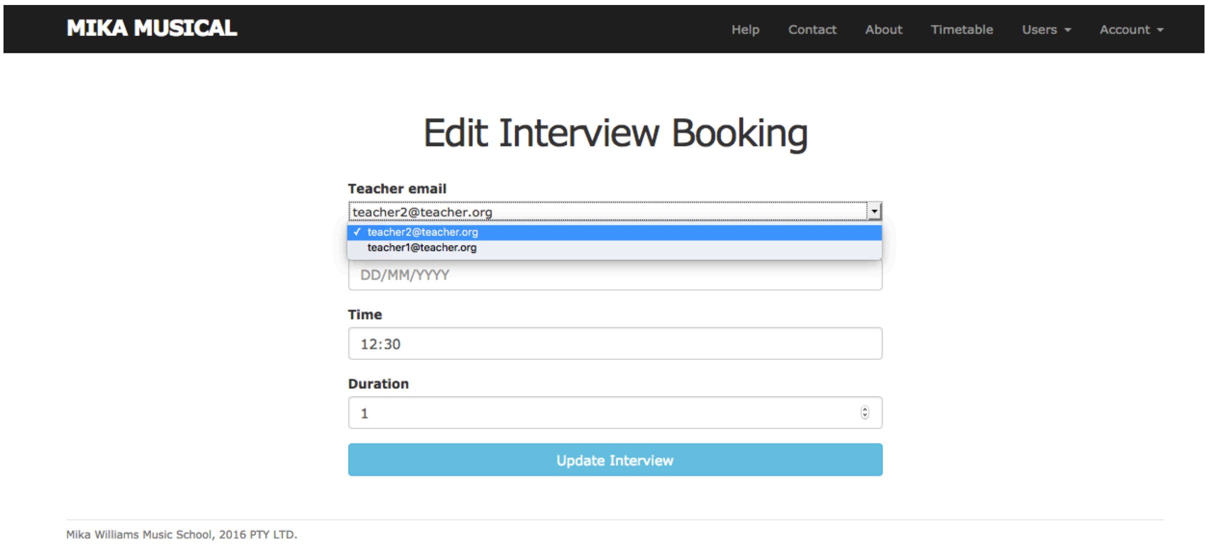
Interview Booked!

Your Current Interviews

Teacher Email	Date	Time	Duration	
teacher2@teacher.org	12/12/2016	12:30	1	Edit Delete
teacher1@teacher.org	22/11/2016	08:30	2	Edit Delete

Mika Williams Music School, 2016 PTY LTD.

Figure 13. Index View for Interviews (On Successful Add)



Edit Interview Booking

Teacher email

teacher2@teacher.org

✓ teacher2@teacher.org

teacher1@teacher.org

DD/MM/YYYY

Time

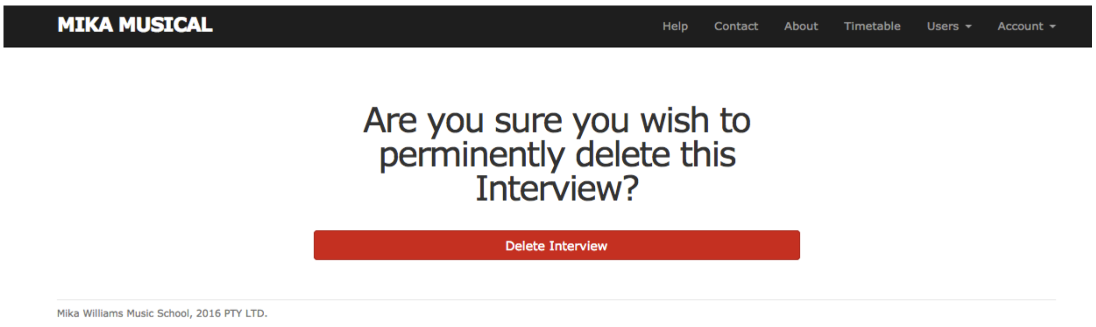
12:30

Duration

1

Mika Williams Music School, 2016 PTY LTD.

Figure 14. Edit View for Interviews



Are you sure you wish to perminently delete this Interview?

Mika Williams Music School, 2016 PTY LTD.

Figure 15. Delete View for Interviews

All views for the modules added follow the layout and structure shown above. Each module is almost identical as previously stated. Small additions such as button colour and table layout were tweaked in order to make the website look more complete and professional. The collection selects in the case shown above only allows the admin to select from the teacher emails located in the database.

Final Testing:

After the completion of all the modules I went through and added, edited and deleted information into all the modules. Small errors in the code were found and patched as required. However, at this stage of development, all modules work accordingly. It is planned to have test fixtures and integration tests added to test all of the modules. This will allow future development to be tested incrementally as additions are added. Furthermore, some design aspects of the form will most likely be changed in future development to make each form look more professional.