1.create project starter

2. setup database

3. add jetstream/livewire :

composer require laravel/jetstream

php artisan jetstream:install livewire –teams

4. migrate tables to database:

Php artisan migrate

5. Create Models and migration :

* Page:
  + Title : string , nullable
  + Slug : string, nullable
  + Content: longtext , nullable
  + Create livewire components :
    - Php artisan make:livewire Pages
    - Php artisan make:livewire FrontPage
    - Add Trix editor :
      * <https://github.com/amaelftah/laravel-trix>

6. Setup Routes:

-create group route to access the pages through authentification middleware

-create modal and bind variables to it

7. Pages Controller:

* Create modal for article creation/modification
* Create modal for delete
* Bind modal for each case to a variable
* Change the value of the variable when a button clicked for the specific action
* Bind the variables in controller to modals then reset them after validation, submission of action

8. FrontPage:

Like single page application the frontend is one page that calls specific components when needed and call specific data from database

9. Create Navigation system for front end using tailwindcss / alpine.js

10. Create Navigation menus controller:

* Create modal for navigation menu creation and update
* Attach logic and variable bind the variables
* Do the same for the delete

11. User Access Management

* We add a column to user table to specify role : *php artisan make:migration add\_role\_to\_users\_table --table=users*
* create middleware to manage access : *php artisan make:middleware AccessByRole*
* add condition for each user role to access specific pages
* Add middleware routedMiddleware to kernel
* Create table in database to add/delete user’s permissions

12. Command to generate component:

* Create a customized command to generate a component from blueprint
* Create two basic blade views similar to admin/pages for users and user-permissions
* Create model and migration for user-permissions: *php artisan make:model UserPermission --migration*
  + Add two string nullable columns “role” and “route\_name”
  + Migrate to database
* To create command for generating our components we need to customize the stub:
  + Generate the stubs : php artisan stub:publish
  + Create command using: php artisan make:command CommandName
  + Put logic of the template Class and ClassModelName in Commands/CommandName.php
  + Create and edit stub that will be used by this CommandName class to generate our class and view
  + Create products/services and announces crud

13. integrate frontend:

* Transform adobe xd design to html/css => integrate to blade templates in the app
* Build components based on the logic in the draft
* Use conditions to render specific components based on page title

14. Bookings :

1. Create bookings controller BookingController , and bookings view folder
2. Work on database :
   1. Employee Model and migration (simply name)
   2. Service model and migration (name: string, duration: int)
      1. Add duration to modal in backend and controller
   3. Many to many relationship between employee and service
   4. Create pivot table to link employee and services
   5. Create schedule model with migration (date,starts\_time,end\_time)
3. Create timeSlotGenerator class that will be used to generate slots depending on schedule and service tables and taking filters into consideration
4. Create filter for slots:
   1. Create filter interface
   2. Implements SlotsPassedToday using slotsPassedToday interface using carbonPeriod
5. Create schedule unavailabilities
   1. Model and migration (start and end time of unavailability)
   2. Cast the properties to be used in Carbon methods
   3. add relationships to schedule
   4. Add new filter by unavailabilities of employees
6. Create appointments table :
   1. Model and migration : add uuid\* (universally unique identifier) and token to make each appointment unique), employee\_id, user\_id,service\_id
   2. Create appointment filter to remove booked slots
   3. Call existing appointments in the controller to remove it so user can not book it
7. Refactor Bookingcontroller:
   1. Move slots code to create booking into Employee method called availableSlots
   2. Start UI :
      1. Component for creating booking
      2. Wire state variable of type array to contains all variables
      3. Create a calendar components
      4. Emits time from the calendar to the create booking component controller
      5. Add validation from the form
      6. bind the UI to create booking
      7. add create booking method that takes the data from form and from the other models and store them in the appointment table
      8. add cancelling option using eloquent scope

\*A universally unique identifier (UUID) is **a 128-bit label used for information in computer systems**. The term globally unique identifier (GUID) is also used, often in software created by Microsoft. When generated according to the standard methods, UUIDs are, for practical purposes, unique.

Issues:

* Add bookings admin view for version beta just looking into the views
* Work on pagination for frontend (optional needed in case of more products need to look into livewire solutions
* Work on references => video youtube (done)
* Fix minor bag in booking // done => modified time zone in the app and php.ini
* Work on front end for booking (done)
* Work on UML for appointment and diagram sequences (on going)

Stack:

Laravel

Livewire

TailwindCss

Alpine.js (tested not used)

Bootstrap