Earn Deployment Guide

Summary

The Earn application is the first prototype of a sales commission calculator web application for beautician teams. This application has been designed with modularity in mind for future expansion with frameworks and a DSL in place for future developers to continue work on this open-source project. This prototype will make up a small piece of a more extensive clinic management suite.

Development Environment Deployment

System Requirements

The following libraries must first be installed on either a Linux or Windows system for application deployment.

- PHP >7.3.
- Composer >2.0.11
- NPM >6.13.4

Linux - Library Installation

Each of the libraries can be installed using the following commands

```
apt install php
apt install npm
apt install composer
```

Windows – Library Installation

On windows each of the libraries are to be installed on the system from their respective websites.

1. NPM - https://nodejs.org/en/download/
2. Composer - https://getcomposer.org/download/

PHP can be installed in conjunction with the SQL server using the guide referenced in the following section, available here.

Database

The database is hosted using a **SQL Server.** There are many options for the SQL server and client. An option for deployment is to follow this guide that works for both Windows and Linux. This guide will install MySQL powered by an Apache server with access to the database through phpMyAdmin. Other SQL options include but are not limited to:

- MySQL Workbench
- MariaDB
- Oracle

Application Deployment

1. From the repository here either download a zip folder of the repository or nn a command prompt in the desired directory, clone the GitHub repository to your local machine
   ```
   git clone https://github.com/any-code/earn.git
   ```

2. Inside of the application for a windows deployment run the following commands in their respective order.
   ```
   npm install
   composer update
   ```

   For a Linux deployment, the php unit library must be installed for the composer command to function correctly.

   ```
   npm install
   apt install phpunit
   composer update
   ```

   These commands will install all of the required libraries specific in the composer.json & package.json files.

3. In the top file directory locate and make a copy of the *.env.example* file renamed as *.env*

4. In the .env, lines 10-15 manage the database connection. If following the SQL database installation guide as specified above, these lines do not need modification. If using your own database server, modify the fields accordingly.

5. Inside your database server create a new database named *earn_app* as specified on line 13 of the .env file. You can alternatively modify this to a name of your choosing.

6. Inside your top-level app directory run the following command to migrate the applications models into your database.
   ```
   php artisan migrate
   ```

7. Inside your top-level app directory generate your application keys for the .env file using the following command.
   ```
   php artisan key:generate
   ```

8. A mail server must be configured to create user accounts and invite team members. This step is not required to use the rest of the applications functionality in a development environment though. To just use mock users, skip to step 12 and see the 'Mock Data' section.

9. Create an email account with a provider of your choice. For this guide create a Gmail account.

10. Inside the .env file, lines 29-36 are for the email service provider. If using Gmail, the only fields that need modification are MAIL_USERNAME, MAIL_PASSWORD, MAIL_FROM_ADDRESS. Complete these fields appropriately for your new email account.

11. For different mail providers other fields will need modification with details provided by the mail service provider.

12. The application can now be deployed in a development environment with the command.

    ```
    php artisan serve
    ```

Mock Data

This deployment can be used to test system functionalities however will lack the ability to retrieve invoices from Xero services as this requires an account actively producing data from their invoice API. To generate mock invoices and users two console commands have been written.

```
php artisan populate:db X Y
```

This command will create a manager's team and then populate the database based on the X and Y paramaters.

- X = The number of employees to add to a managers team
- Y = The number of sales to randomly allocate to employees, each sale backdating 1 day from the current day. A rule is created that backdates from the current date to the oldest sale.

```
php artisan populate:sales X
```

This command can either be used in two ways. It can be run subsequently to the above command, applying further sales randomly to all employees of the manager team. If using a fresh database, the command will create one manager and one employee, with all sales being assigned to the one employee. It is worth noting that this command applies commission based on the currently existing rules, so from a fresh database you may also want to define a commission rule otherwise no commission will be paid.

- Y = The number of sales to randomly allocate to employees, each sale backdating 1 day from the current day.

See further sections for integration of mail server and Xero invoice services.

Xero

For development purposes, the deployment of the application will retrieve sales data and create invoices from an invoice & sales generator that runs through the invoice parser.

The infrastructure for the invoice parser to be used with Xero is in place and can only be used with an **active Xero company account with invoices**. This section describes how to switch the invoice parser from mock data to Xero data. This guide presumes an active Xero account is registered.

1. Create the Earn application within Xero Developer, filling out details as appropriate to your business.

2. When completing the field 'Redirect URIs', while entering your appropriate deployment URL, use the subdirectory ''http://YOUR_WEBSITE/login/xero''.

3. One the same application details page, generate a client secret below the client id number.

4. Enter your client id, client secret and redirect URI in the .env file on lines 51-53.

5. Because our client’s Xero configuration was very specific and non-standard, you will need to implement a fetcher class that adheres to the interface defined in ‘app\Classes\InvoiceFetcherInterface.php’.

6. Inside of the application file ' app\Http\Controllers\ParsingController.php', add your fetcher to the imports and add it to the $fetchers dictionary defined on line 76.

7. Start the application and log in as either a manager or employee for the team with desired xero access.

8. Access the team settings and select log into xero for in the 'Available services' section.

9. Fill out your xero credentials and you will be returned to the application with the Xero token now stored in the database.

10. Further information regarding available methods for interacting with Xero data is available here