

## WEB DEVELOPMENT USING LARAVEL AND XAMPP

Website development would require a developer to understand the concepts of HTML (Hypertext Markup Language), CSS, and Javascript. These three components make up the pages and presentations that will be viewed by a user or rather called as Front-End side of a website.

Meanwhile, the server-side programming (Back-End side of a website) allows the interaction with the database to retrieve information and processing information that will later be displayed to the users. The Back-End side of a website may involve SQL and PHP. PHP is a programming language used for web development. PHP scripts may also include HTML and Javascript codes. SQL (Structured Query Language) is a language that allows the execution of queries to retrieve, insert, update and delete data from/to a database. MySQL for example, is a Relational Database Management System where it stores database objects in the form of tables that can be interconnected to each other.

XAMPP is software for a web development where it contains the configuration of Apache, PHP and MySQL. XAMPP is a useful platform that can be used to develop and test a website in the testing server (localhost) before releasing it to the main server. XAMPP is an abbreviation for cross-platform (X) which means that the software can be installed in various platforms including Windows, Linux and Mac OS X operating system; Apache (A) which is a web server software that can run a website such as WordPress and others; MySQL (M) which is a database management software which manages database objects, PHP (P) which is a programming language used for server-side programming; Perl (P) which is an object-oriented programming language for general purposes.

Laravel is a web application framework that builds application using PHP with designated syntax. Laravel has several components: i) Model (connects to the database which allows retrieval of data), ii) Controller (compiles requests and actions) and (iii) Views (render pages on the front-end side of the web). Additionally, routes are used to map URLs to controller actions.

Laravel is best used to build a website that is based on CRUD operations – create, read, update, and delete resources available on the website. Here, we will build a simple web application/database using XAMPP and Laravel version 5.8. Note that the steps in this guideline are performed on Mac OS X (Catalina). The installation steps and directories could be different according to the operating systems of your computer.

The data used for database generation in MySQL are obtained from DrugBank, which includes information related to approved drugs and their targets.

### Installation

---

#### *1. XAMPP installation*

---

The latest version of XAMPP (version [7.4.20](#) as of June 2021) can be downloaded from [here](#). XAMPP is equipped with its version of Apache, MySQL, PHP and Perl. Run the .dmg or .exe file and follow the instructions for installation. XAMPP will be stored under /Applications/ (in Mac) or C:\ (in Windows).

---

## 2. Code Editor (IDE)

---

Code editor can be used to create, read, or edit the code. The Xcode IDE facilitates the Mac OS X users to edit and read their codes through its simple interface. It is free and can be downloaded from the Apple Store. There are several options of IDE for Mac OS X, Linux and Windows, which can be explored [here](#).

---

## 3. Composer

---

Composer is a tool for PHP dependency management, where it manages the dependencies or libraries needed for your project. Further information on the composer can be obtained [here](#). To install composer, type the following commands in the terminal:

```
# Get a copy of PHP setup file to the local directory
$ php -r "copy('https://getcomposer.org/installer','composer-setup.php');"
$ php -r "if (hash_file('sha384','composer-setup.php') ===
'756890a4488ce9024fc62c56153228907f1545c228516cbf63f885e036d37e9a59d27d63f46af1
d4d07ee0f76181c7d3') { echo 'Installer verified'; } else { echo 'Installer corrupt';
unlink('composer-setup.php'); } echo PHP_EOL;"

# Run the setup file
$ php composer-setup.php
$ php -r "unlink('composer-setup.php');"

# Move the file to executable path
$ sudo mv composer.phar /usr/local/bin/composer

# Run composer to test whether it works or not
$ composer
```

---

## 4. Install Laravel on localhost XAMPP

---

Laravel is a PHP framework for web development where it follows the model-view-controller (design) structure. Here, Laravel project will be created in the XAMPP htdocs folder. In Mac, the folder could be here: /Applications/XAMPP/xamppfiles/htdocs/, while in Windows, the folder could be C:\XAMPP\htdocs\ . The latest version of Laravel will be installed (version 8.5.20 as of June 2021). Further information on Laravel can be obtained [here](#).

The Laravel project will be created using composer using the following command:

```
# Go to the htdocs directory
$ cd /Applications/XAMPP/xamppfiles/htdocs/

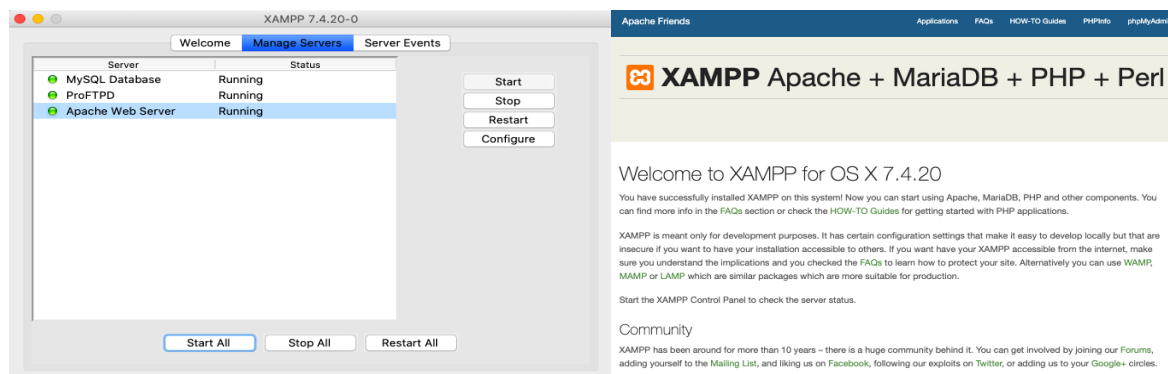
# Create a new Laravel project called myproject.
# The folder 'myproject' will be stored under htdocs directory
$ composer create-project --prefer-dist laravel/laravel myproject

#OR to install a specific version of Laravel (e.g. Laravel 5.8)
$ composer create-project --prefer-dist laravel/laravel myproject "5.8.*"
```

## Configuration

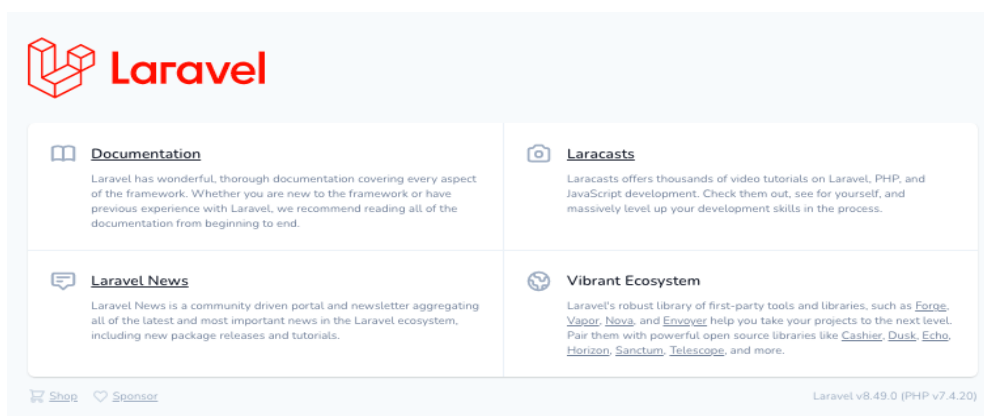
### 1. Run XAMPP

Start the XAMPP application or control panel (manager-osx). Under 'Manager Servers' tab, click 'Start All' to start all modules: MySQL Database, ProFTPD, and Apache Web Server. The status for all modules will be changed to 'Running' (or shown in green color) indicating that all modules have been started successfully. Browse to localhost/ (or <http://127.0.0.1>) and a welcome page will be shown, indicating that the XAMPP is working fine.



### 2. Open Laravel project in the web browser

To open the Laravel project (myproject) on the web, browse to [localhost/myproject/public/](http://localhost/myproject/public/). An index page will be shown.



To change the link from localhost/myproject/public to localhost/myproject, use the following commands:

```
# Go to myproject directory
$ cd /Applications/XAMPP/xamppfiles/htdocs/myproject

# Rename server.php in your Laravel root folder to index.php
$ mv server.php index.php

# Copy the .htaccess file from /public directory to myproject root folder
$ cp public/.htaccess ./
```

Now, myproject can also be accessed from <http://localhost/myproject/>.

---

### 3. Connecting Laravel project to MySQL database

---

*Step 1: Create a new database via phpMyAdmin or through command line*

Here, a database named 'drugdb' will be created using sample data from DrugBank. The database will consist of three tables; (i) approved\_drugs, (ii) drug\_targets and (iii) target\_sequences.

Table	List of columns
1. approved_drugs	<b>drugbankid</b> , drugname, drugtype, indication, pubchemid, hetid
2. drug_targets	pharmaactive, proteinname, genename, gbproteinid, gbgeneid, <b>uniprotid</b> , uniprottitle, pdbids, species, <b>drugids</b>
3. target_sequences	moleculetype, <b>uniprotid</b> , name, <b>drugids</b> , sequence

\*The bold text indicates ids that will be used to interconnect information across tables.

The phpMyAdmin web interface can be used to create a database in MySQL by browsing to <http://localhost/phpmyadmin/>. In this case, the database will be created using the command line. 'mysql' command from XAMPP will be used (/Applications/XAMPP/xamppfiles/mysql).

First, download and store the CSV-formatted files that will be used to generate the database into the 'db' folder in myproject directory. The CSV file contains truncated data from the original table retrieved from DrugBank.

Truncated version:

[https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/approved\\_drugs.csv](https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/approved_drugs.csv)

Truncated version:

[https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/drug\\_targets.csv](https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/drug_targets.csv)

Truncated version:

[https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/target\\_sequences.csv](https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/target_sequences.csv)

```
# Create a folder named 'db' inside myproject root directory
$ cd /Applications/XAMPP/xamppfiles/htdocs/myproject
$ mkdir db

# Download the data and store it in 'db' folder using curl or wget command
$ curl -LJ -o db/approved_drugs.csv
https://raw.githubusercontent.com/NurSyatila/XAMPP Laravel/main/sampledata/approved_drug
s.csv

$ curl -LJ -o db/drug_targets.csv
https://raw.githubusercontent.com/NurSyatila/XAMPP Laravel/main/sampledata/drug_targets.cs
v

$ curl -LJ -o db/target_sequences.csv
https://raw.githubusercontent.com/NurSyatila/XAMPP Laravel/main/sampledata/target_sequenc
es.csv
```

Enter the MySQL interface using MySQL command from XAMPP and create the database.

```

# Enter MySQL
# By default, the username and password are 'root' and ''
$ /Applications/XAMPP/xamppfiles/bin/mysql -u root -p (press Enter when 'Enter password'
argument popup)

# MySQL monitor will be displayed
# show available databases
> SHOW DATABASES;

# create a database named 'drugdb', CREATE DATABASE {database name}
> CREATE DATABASE drugdb;
> USE drugdb;

# create tables, CREATE TABLE {Table name} ({Columns})

# a) create approved_drugs table according to the column name in CSV file
> CREATE TABLE approved_drugs (
    drugbankid TEXT DEFAULT NULL,
    drugname TEXT DEFAULT NULL,
    drugtype TEXT DEFAULT NULL,
    indication TEXT DEFAULT NULL,
    pubchemid TEXT DEFAULT NULL,
    hetid TEXT DEFAULT NULL);
# import the CSV file to be included as rows of the table
> LOAD DATA LOCAL INFILE 'db/approved_drugs.csv' INTO TABLE approved_drugs FIELDS
TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

# b) create drug_targets table according to the column name in CSV file
> CREATE TABLE drug_targets (
    pharmaactive TEXT DEFAULT NULL,
    proteinname TEXT DEFAULT NULL,
    genename TEXT DEFAULT NULL,
    gbproteinid TEXT DEFAULT NULL,
    gbgeneid TEXT DEFAULT NULL,
    uniprotid TEXT DEFAULT NULL,
    uniprottitle TEXT DEFAULT NULL,
    pdbids TEXT DEFAULT NULL,
    species TEXT DEFAULT NULL,
    drugids TEXT DEFAULT NULL);
# import the CSV file to be included as rows of the table
> LOAD DATA LOCAL INFILE 'db/drug_targets.csv' INTO TABLE drug_targets FIELDS
TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

# c) create target_sequences table according to the column name in CSV file
> CREATE TABLE target_sequences (
    moleculetype TEXT DEFAULT NULL,
    uniprotid TEXT DEFAULT NULL,
    name TEXT DEFAULT NULL,
    drugids TEXT DEFAULT NULL,
    sequence TEXT DEFAULT NULL);
# import the CSV file to be included as rows of the table
> LOAD DATA LOCAL INFILE 'db/target_sequences.csv' INTO TABLE target_sequences FIELDS
TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';

# show tables in database drugdb that have been generated
> SHOW TABLES;

#exit MySQL monitor
> exit

```

Use phpMyAdmin web interface to view the structure and content of individual tables in drugdb database (<http://localhost/phpmyadmin>). In phpMyAdmin, click 'drugdb' from the list of all databases on the left panel of the window.

The screenshot shows the phpMyAdmin web interface. On the left, a sidebar lists databases: New, drugdb, New, approved\_drugs, drug\_targets, target\_sequences, information\_schema, mysql, performance\_schema, phpmyadmin, and test. The main panel shows the 'drugdb' database selected. A table list displays three tables: 'approved\_drugs' (290 rows), 'drug\_targets' (50 rows), and 'target\_sequences' (114 rows). A summary row shows '3 tables' with a total of '454' rows.

Table	Action	Rows	Type	Collation
approved_drugs	Browse Structure Search Insert Empty Drop	290	InnoDB	utf8mb4_0900_ai_ci
drug_targets	Browse Structure Search Insert Empty Drop	50	InnoDB	utf8mb4_0900_ai_ci
target_sequences	Browse Structure Search Insert Empty Drop	114	InnoDB	utf8mb4_0900_ai_ci
3 tables	Sum	454	InnoDB	utf8mb4_0900_ai_ci

(List of tables available in drugdb database)

The screenshot shows the 'approved\_drugs' table selected. The SQL query is 'SELECT \* FROM `approved\_drugs` ORDER BY `approved\_drugs`.`hetid` DESC'. The table has 25 rows displayed. The columns are: drugbankid, drugname, drugtype, indication, pubchemid, and hetid.

drugbankid	drugname	drugtype	indication	pubchemid	hetid
DB00909	Zonisamide	SmallMoleculeDrug	For use as adjunctive treatment of partial seizure...	46505278	ZON
DB00477	Chlorpromazine	SmallMoleculeDrug	For the treatment of schizophrenia; to control nau...	46508395	Z80
DB09079	Nintedanib	SmallMoleculeDrug	In the US, nintedanib is indicated for the treatme...	310265007	XIN
DB00170	Menadiione	SmallMoleculeDrug	The primary known function of vitamin K is to assi...	46505447	VK3
DB00165	Pyridoxine	SmallMoleculeDrug	Pyridoxine is indicated for the treatment of vitam...	46508560	UEG
DB09270	Ubidecarenone	SmallMoleculeDrug	The diet supplements containing ubidecarenone are ...	310265165	U10
DB00316	Acetaminophen	SmallMoleculeDrug	In general, acetaminophen is used for the treatmen...	46506142	TYL

(Table 1: approved\_drugs)

The screenshot shows the 'drug\_targets' table selected. The SQL query is 'SELECT \* FROM `drug\_targets`'. The table has 50 rows displayed. The columns are: pharmaactive, proteinname, genename, gbproteinid, gbgeneid, uniprotid, uniprottitle, pdbids, species, and drugids.

pharmaactive	proteinname	genename	gbproteinid	gbgeneid	uniprotid	uniprottitle	pdbids	species	drugids
P	Peptidoglycan synthase FtsI	ftsI	1574687	L42023	P45059	FTSI_HAEIN		Haemophilus influenzae (strain ATCC 51907 / DSM 11...	DB00303
N	Histidine decarboxylase	HDC	32109	X54297	P19113	DCHS_HUMAN	4E1O	Humans	DB00114
N	Glutaminase liver isoform, mitochondrial	GLS2	6650606	AF110330	Q9UI32	GLSL_HUMAN	4BQM	Humans	DB00142

(Table 2: drug\_targets)

moleculetype	uniprotid	name	drugids	sequence
protein	P45059	Peptidoglycan synthase FtsI	DB00303	MVKFNSSRKSGKSKKTIRKLTAPETVKQNKPKVFEKCFMRGRYMLSTVL...
protein	P19113	Histidine decarboxylase	DB00114	MMEPEEYRERGREMVDYICQYLSTVRERRVTPDVQPGYLRQLPESAPED...
protein	Q9UI32	Glutaminase liver isoform, mitochondrial	DB00142	MRSMKALQKALS RAGSHCGRGGWGHPSRSPLLGGGVRRHHLSEAAQGRET...
protein	P00488	Coagulation factor XIII A chain	DB11300; DB11311; DB11571; DB13151	MSETSRATFGGRRVPPNNSNAEDDLPTVELQGQVPRGVNLQEFNLNVT...

(Table 3: target\_sequences)

## Step 2: Configure database connection and settings

### a) configure database connection in .env configuration file

Open the .env file located in the myproject directory, and change the DB\_DATABASE to match the generated database 'drugdb'. If applicable, change the DB\_USERNAME and DB\_PASSWORD.

```
DB_CONNECTION=mysql
DB_HOST=190.0.0.1
DB_PORT=4008
DB_DATABASE=drugdb
DB_USERNAME=root
DB_PASSWORD=
```

### b) configure a database connection in myproject/config/database.php

Open the database.php file located in myproject/config directory and change the 'database', 'username' and 'password'.

```
'mysql' => [
    'driver' => 'mysql',
    'url' => env('DATABASE_URL'),
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', 'drugdb'),
    'username' => env('DB_USERNAME', 'root'),
    'password' => env('DB_PASSWORD', ''),
    'unix_socket' => env('DB_SOCKET', ''),
    'charset' => 'utf8mb4',
    'collation' => 'utf8mb4_unicode_ci',
    'prefix' => '',
    'prefix_indexes' => true,
    'strict' => true,
    'engine' => null,
    'options' => extension_loaded('pdo_mysql') ? array_filter([
        PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
    ]) : [],
],
```

Use the 'migrate' artisan command to connect the database to Laravel. Here, PHP from XAMPP will be used to prevent errors resulting from the use of different PHP versions.

```
#clear the cache after changing the configuration
$ /Applications/XAMPP/xamppfiles/bin/php artisan cache:clear
# run migration to update the database
$ /Applications/XAMPP/xamppfiles/bin/php artisan migrate
```

---

#### 4. Configure Model, View and Controllers

---

##### Step 1: Configure database and model (myproject/app/Models)

Model is an entity in Laravel that communicates with the database. A model represents a connection to a single table in the database. In this case, we have three tables: approved\_drugs, drug\_targets, and target\_sequences. The make:model artisan command can be used to create models for individual tables.

\*Note that PHP from XAMPP will be used to run artisan command in Laravel.

```
#create model for individual tables
$ /Applications/XAMPP/xamppfiles/bin/php artisan make:model ApprovedDrugs
$ /Applications/XAMPP/xamppfiles/bin/php artisan make:model DrugTargets
$ /Applications/XAMPP/xamppfiles/bin/php artisan make:model TargetSequences
```

The models are stored in myproject/app/Models/. View the file and manually edit the model to specify the table that will be used for a particular model.

<pre>1 &lt;?php 2 3 namespace App\Models; 4 5 use Illuminate\Database\Eloquent\Model; 6 7 class ApprovedDrugs extends Model 8 { 9     protected \$table = 'approved_drugs'; 10 }</pre>	<pre>1 &lt;?php 2 3 namespace App\Models; 4 5 use Illuminate\Database\Eloquent\Model; 6 7 class DrugTargets extends Model 8 { 9     protected \$table = 'drug_targets'; 10 }</pre>	<pre>1 &lt;?php 2 3 namespace App\Models; 4 5 use Illuminate\Database\Eloquent\Model; 6 7 class TargetSequences extends Model 8 { 9     protected \$table = 'target_sequences'; 10 }</pre>
--	--	--

##### Step 2: Configure views (resources/views/), routes (routes/web.php) and controllers (app/Http/Controllers/)

The web.php in routes directory defines all routes for web interfaces that are connected to PHP files from the view entity in Laravel. Route can include the following functions: simple function call, redirect from a route to another URI, the API / query using GET and POST methods with and without parameters.

Views contain the HTML or PHP codes that are stored with prefix blade.php in resources/views directory. Views serve the front-end side of the website.

The controller organizes request handling in the form of functions. Controllers are stored in myproject/app/Http/Controllers/. The make:controller artisan command can be used to create a controller file. As a start, we can create a single controller that communicates with the tables in drugdb, extract necessary information and produce a view containing the list of the information we needed.

The default homepage for myproject website is located at resources/views/welcome.blade.php. The welcome.blade.php can be edited to fit the purpose of the website, such as to include a query to a selected table in the database or to include a link to another section.



```

65     </head>
66     <body>
67         <div class="flex-center position-ref full-height">
68             <div class="content">
69                 <div class="title m-b-md">
70                     DrugDB
71                 </div>
72
73                 <div class="links">
74                     <a href="{{ url('/') }}">Home</a>
75                     <a href="{{ url('section/approveddrugs') }}">Approved Drugs</a>
76                     <a href="{{ url('section/drugtargets') }}">Drug Targets</a>
77                     <a href="{{ url('section/targetsequences') }}">Target Sequences</a>
78                 </div>
79
80             </div>
81         </div>
82     </div>
83 </body>
84 </html>

```

(resources/views/welcome.blade.php)

# DrugDB

HOME

APPROVED DRUGS

DRUG TARGETS

TARGET SEQUENCES

(The view from the web browser)

In welcome.blade.php, we manipulate the file to show links to sections for APPROVED DRUGS, DRUG TARGETS and TARGET SEQUENCES.

Now, let's create a page for individual sections which can be viewed in the form of a table. To do so, we have to create view pages for individual sections and a single controller that can store the function to call relevant tables and view the individual sections.

\*Note that it is not recommended to create a view containing a list of all data from the MySQL database (especially for a very large database). The page might return an error due to the inability to process a large database. This is just for a learning purpose.

Type	Files
View	section/approveddrugs.blade.php (to view list containing approved drugs) section/drugtargets.blade.php (to view list containing drug targets) section/targetsequences.php (to view list containing sequences)
Controller	SectionsController.php (to create action for individual sections)

## CONTROLLER

Here, we will create a controller that can retrieve all rows from a table.

To create the controller, use the following command:

```

# create a controller - will be saved in app/Http/Controllers
$ /Applications/XAMPP/xamppfiles/bin/php artisan make:controller SectionController

```

A file named SectionController.php will be created. Manually edit the file to include the function to retrieve information from the database table. Here, three functions are added to retrieve individual tables. For example, the drugsection() function will store the data from table 'approved\_drugs' as 'drugs' and links the function to section/approveddrugs view. No filtering of the data is performed here. The filtering and sorting of MySQL table in Controller will be further discussed in the following sections. Note that the models and relevant modules are invoked at the beginning of the scripts.

```
1  <?php
2
3  namespace App\Http\Controllers;
4  //Include models that will be used in this controller
5  use App\Models\ApprovedDrugs;
6  use App\Models\DrugTargets;
7  use App\Models\TargetSequences;
8  use Illuminate\Http\Request;
9  use Illuminate\Support\Facades\DB;
10
11 class SectionController extends Controller
12 {
13     public function drugsection(){
14         $drugs = ApprovedDrugs::all();
15         return view('section.approveddrugs', compact('drugs'));
16     }
17
18     public function targetsection(){
19         $drugs = ApprovedDrugs::all();
20         return view('section.drugtargets', compact('targets'));
21     }
22
23     public function sequencesection(){
24         $sequences = DB::table('target_sequences')->get();
25         return view('section.targetsequences', compact('sequences'));
26     }
27 }
```

On the other hand, the command `php artisan make:controller {name}Controller --resource` can be used to quickly generate controller with CRUD (create, read, update, delete) actions.

## VIEW

Create files under resources/views/section directory. The files should contain PHP and HTML codes that generate the table view. View will use data from Controller to be presented to the user. For example, the section/approveddrugs view extracts the 'drugs' variable from the SectionController and presents the data as a table. Additionally, CSS can be used to improve the visualization of the data. DataTable, which is a jquery plug-in will be used to represent data in a tabulated form that allows filtering and searching. To use DataTable, relevant links to the jquery scripts should be included in blade PHP under views. Further information on DataTable can be obtained [here](#).

<pre> 90 &lt;center&gt;&lt;div style="width: 90%; align="center"&gt; 91 &lt;center&gt;&lt;br&gt;&lt;b&gt;List of approved drugs from DrugBank (truncated)&lt;/b&gt;&lt;/center&gt; 92 &lt;table id="mytable" class="display compact" style="width:100%"&gt;&lt;thead&gt;&lt;tr&gt; 93   &lt;th&gt;DrugBank ID&lt;/th&gt; 94   &lt;th&gt;Drug name&lt;/th&gt; 95   &lt;th&gt;Drug Type&lt;/th&gt; 96   &lt;th&gt;Indication&lt;/th&gt; 97   &lt;th&gt;PubChem ID&lt;/th&gt; 98   &lt;th&gt;HET ID&lt;/th&gt; 99 &lt;/tr&gt;&lt;/thead&gt; 100 &lt;tbody&gt; 101 @foreach(\$drugs as \$d) 102 &lt;tr&gt; 103   &lt;td&gt;{{ \$d-&gt;drugbankid }}&lt;/td&gt; 104   &lt;td&gt;{{ \$d-&gt;drugname }}&lt;/td&gt; 105   &lt;td&gt;{{ \$d-&gt;drugtype }}&lt;/td&gt; 106   &lt;td&gt;{{ \$d-&gt;indication }}&lt;/td&gt; 107   &lt;td&gt;{{ \$d-&gt;pubchemid }}&lt;/td&gt; 108   &lt;td&gt;{{ \$d-&gt;hetid }}&lt;/td&gt; 109 &lt;/tr&gt; 110 @endforeach 111 &lt;/tbody&gt; 112 &lt;tfoot&gt;&lt;tr&gt; 113   &lt;th&gt;DrugBank ID&lt;/th&gt; 114   &lt;th&gt;Drug name&lt;/th&gt; 115   &lt;th&gt;Drug Type&lt;/th&gt; 116   &lt;th&gt;Indication&lt;/th&gt; 117   &lt;th&gt;PubChem ID&lt;/th&gt; 118   &lt;th&gt;HET ID&lt;/th&gt; 119 &lt;/tr&gt;&lt;/tfoot&gt; 120 &lt;/table&gt;&lt;br&gt; 121 &lt;/div&gt;&lt;/center&gt; </pre>	<pre> 125 &lt;script&gt; 126 127 \$(document).ready(function() { 128   // Setup - add a text input to each footer cell 129   \$('#mytable tfoot th').each( function () { 130     var title = \$(this).text(); 131     \$(this).html( 'input type="text" placeholder="Search" /&gt;' ); 132   } ); 133 134   // DataTable 135   var table = \$('#mytable').DataTable({ 136     initComplete: function () { 137       // Apply the search 138       this.api().columns().every( function () { 139         var that = this; 140 141         \$( 'input', this.footer() ).on( 'keyup change clear', function () { 142           if ( that.search() !== this.value ) { 143             that 144               .search( this.value ) 145               .draw(); 146           } 147         } ); 148       } ); 149     } 150   }); 151 152 } ); 153 154 &lt;/script&gt; </pre>
(Define the table columns and rows)	(Datatable javascript used to generate table)

## ROUTES

Routes can be defined in routes/web.php. '/' represents the homepage or welcome page (localhost/myproject/). A new folder has been created which stores files for individual sections that represent individual tables in database 'drugdb'. Clicking the page localhost/myproject/section/approveddrugs will lead to a new page representing the APPROVED DRUGS section. At the beginning of the web.php, the route to Controller is invoked to link the view to the corresponding controller.

```

1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\SectionController;
5
6 /*
7 |-----
8 | Web Routes
9 |-----
10 |
11 | Here is where you can register web routes for your application. These
12 | routes are loaded by the RouteServiceProvider within a group which
13 | contains the "web" middleware group. Now create something great!
14 |
15 */
16
17 Route::get('/', function () {return view('welcome');});
18 // search page that include user interfaces
19 Route::get('search', function () {return view('search');});
20 //sections
21 Route::get('section/approveddrugs', [SectionController::class, 'drugsection']);
22 Route::get('section/drugtargets', [SectionController::class, 'targetsection']);
23 Route::get('section/targetsequences', [SectionController::class, 'sequencesection']);

```

(resources/web.php)

# DrugDB

HOME APPROVED DRUGS DRUG TARGETS TARGET SEQUENCES

List of approved drugs from DrugBank (truncated)

Show 10 entries

Search:

DrugBank ID	Drug name	Drug Type	Indication	PubChem ID	HET ID
DB00277	Theophylline	SmallMoleculeDrug	For the treatment of the symptoms and reversible airflow obstruction associated with chronic asthma and other chronic lung diseases, such as emphysema and chronic bronchitis.	46505949	TEP
DB01229	Paclitaxel	SmallMoleculeDrug	Used in the treatment of Kaposi's sarcoma and cancer of the lung, ovarian, and breast. Abraxane® is specifically indicated for the treatment of metastatic breast cancer and locally advanced or metastatic non-small cell lung cancer.	46506910	TA1
DB00619	Imatinib	SmallMoleculeDrug	For the treatment of Philadelphia chromosome positive chronic myeloid leukemia (Ph+ CML), Ph+ acute lymphoblastic leukaemia, myelodysplasticmyeloproliferative diseases, aggressive systemic mastocytosis, hypereosinophilic syndrome and/or chronic eosinophilic leukemia (CEL), dermatofibrosarcoma protuberans, and malignant gastrointestinal stromal tumors (GIST).	46505055	STI
DB00202	Succinylcholine	SmallMoleculeDrug	Used in surgical procedures where a rapid onset and brief duration of muscle relaxation is needed (includes intubation, endoscopies, and ECT)	46506023	SCK
DB06151	Acetylcysteine	SmallMoleculeDrug	Acetylcysteine is used mainly as a mucolytic and in the management of paracetamol (acetaminophen) overdose.	99443235	SC2
DB00795	Sulfasalazine	SmallMoleculeDrug	For the treatment of Crohn's disease and rheumatoid arthritis as a second-line agent.	46505451	SAS
DB00936	Salicylic acid	SmallMoleculeDrug	Key additive in many skin-care products for the treatment of acne, psoriasis, callouses, corns, keratosis pilaris and warts.	46504942	SAL
DB00162	Vitamin A	SmallMoleculeDrug	For the treatment of vitamin A deficiency.	46508191	RTL
DB00119	Pyruvic acid	SmallMoleculeDrug	For nutritional supplementation, also for treating dietary shortage or imbalance	46505692	PYR
DB01123	Proflavine	SmallMoleculeDrug	Topical antiseptic used mainly in wound dressings.	46505401	PRL

Showing 11 to 20 of 290 entries

Previous 1 2 3 4 5 ... 29 Next

(The view from the web browser)

Now we have successfully created a website called DrugDB which provides information on drugs, targets, and sequences.

## Querying and interconnecting tables

In a comprehensive website, tables from MySQL database are often interconnected to each other, and multiple queries can be performed to get information from different tables.

Several questions can be answered from the query to these interconnected tables in 'drugdb' database include:

- 1) What is the drug that links to a particular ID from other resources?
- 2) Which drugs are indicated for diabetes/cancer? What are the proteins that target these drugs?
- 3) I want to know if my protein/gene is a drug target and which drugs can bind to my protein?
- 4) I want to know if my sequence could be a protein target

Thus, a website may contain an interface that allows users to search for particular information. In this case, the DrugDB may contain an interface that allows users to search for drugs and/or targets given a keyword. It is best to first plan the interfaces or queries that will be included on the website. The homepage could include a search page containing such interfaces.

```
73 <div class="links">
74 <a href="{ url('/') }">Home</a>
75 <!--Include a search page containing user interfaces for querying to the database-->
76 <a href="{ url('search') }">Search</a>
77 <a href="{ url('section/approveddrugs') }">Approved Drugs</a>
78 <a href="{ url('section/drugtargets') }">Drug Targets</a>
79 <a href="{ url('section/targetsequences') }">Target Sequences</a>
80 </div>
```

(search.blade.php)

```

14 Route::get('/', function () {return view('welcome');});
15 // search page that include user interfaces
16 Route::get('search', function () {return view('search');});
17 //sections
18 Route::get('section/approveddrugs', ['uses'=>'SectionController@drugsection', 'as'=>'section.approveddrugs']);
19 Route::get('section/drugtargets', ['uses'=>'SectionController@targetsection', 'as'=>'section.drugtargets']);
20 Route::get('section/targetsequences', ['uses'=>'SectionController@sequencesection', 'as'=>'section.targetsequences']);
21

```

(resources/view.php)

# DrugDB

[HOME](#)
[SEARCH](#)
[APPROVED DRUGS](#)
[DRUG TARGETS](#)
[TARGET SEQUENCES](#)

The DrugDB database allow searching for approved drugs and their targets. All data are obtained from DrugBank.

### SEARCH A DRUG

Search by drug name / DrugBank ID / HET ID / PubChem ID :

Search by drug indication / disease :

### SEARCH A TARGET

Search by drug

(Name / DrugBank ID / HET ID / PubChem ID) :

Search by protein or gene name / ID

(Name / UniProt Title / GeneBank / UniPort / PDB) :

Search by organism :

Search by sequence (fragment or complete sequence of protein / gene)

Insert sequence.

(The view from the web browser)

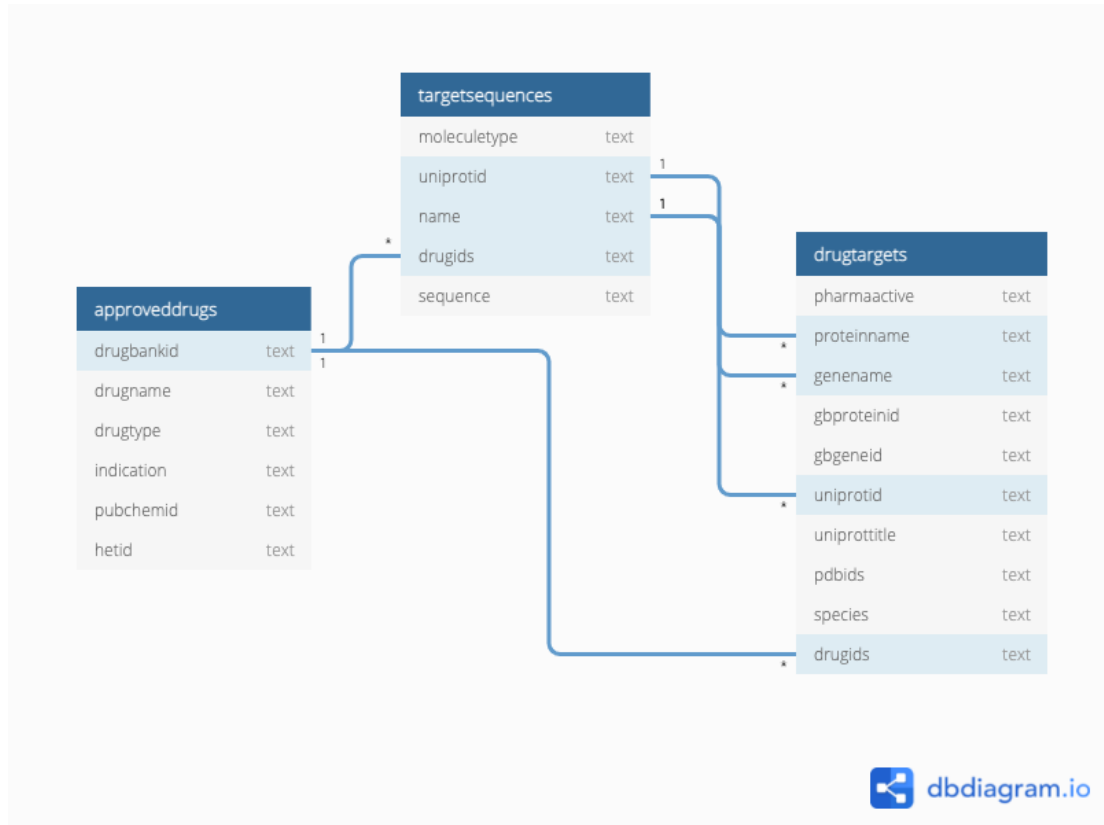
In DrugDB, the interfaces are designed in such a way that users can search for a drug, or a target given certain keywords. The requests from users including the given keywords will be passed to Controller, where the Controller will extract matching information requested by users and pass the values to View.

Query in the web interface	Form method, controller, function	Model and parameters used	View
<b>Search a drug</b>			
Search by drug name / DrugBank ID / HET ID / PubChem ID	M: POST identifier: id C: DrugSearch F: getdrugid	Model: ApprovedDrugs Parameters: drugbankid, drugname, pubchemid, hetid	/drugsearch/ drugid
Search by drug indication/disease	M: POST identifier: indication C: DrugSearch F: getindication	Model: ApprovedDrugs Parameters: indication	/drugsearch/ indication
<b>Search a target</b>			
Search by drug (Name / DrugBank ID / HET ID / PubChem ID)	M: POST identifier: drugid C: TargetSearch F: getdrug	Model: ApprovedDrugs Model: DrugTargets Parameters: <b>drugbankid*</b> , drugname, pubchemid, hetid	/targetsearch/ drugid
Search by protein or gene name / ID (Name / UniProt Title / GeneBank / UniPort / PDB)	M: POST identifier: proteinid C: TargetSearch F: getprotein	Model: DrugTargets proteinname, genename, gbproteinid, gbgeneid, uniprotid, uniprottitle, pdbids	/targetsearch/ protein
Search by organism	M: POST identifier: species C: TargetSearch F: getspecies	Model: DrugTargets Parameter: species	/targetsearch/ organism
Search by sequence (fragment or complete sequence of protein / gene)	M: POST identifier: C: TargetSearch F: getsequence	Model: TargetSequences Model: DrugTargets Parameter: sequence, <b>uniprotid*</b>	/targetsearch/ sequence
<b>Comprehensive view for a drug/target</b>			
Link to information	Form method, controller, function	Model and parameters used	View
Information for a drug	M: GET Identifier: drug (drugbankid) C: DrugSearch F: viewdrug	Model: ApprovedDrugs Model: DrugTargets	/drugsearch/ viewdrug/{drug}
Information for a protein	M: GET Identifier: protein (uniprotid) C: TargetSearch F: viewprotein	Model: TargetSequences	/targetsearch/ viewtarget/{protein}

\*parameter used to build relationship between tables, M: Method, C: Controller, F: Function in controller

The viewdrug and viewtarget page serve as individual pages for given Drugbank ID and Uniprot ID, respectively.

## CONNECTION BETWEEN MODELS



The entity-relationship (ER) diagram has been created using dbdiagram.io

(<https://dbdiagram.io/>)

Diagram link: <https://dbdiagram.io/d/60e270300b1d8a6d3966533a>

## CONTROLLER

Here, two additional controllers will be created, i.e. DrugSearchController.php and TargetSearch.php. The controllers will receive requests/parameters from the user and provide the matching information. In certain cases, output from one table will be used as an input to get output from a different table.

```
# create a controller – will be saved in app/Http/Controllers
$ /Applications/XAMPP/xamppfiles/bin/php artisan make:controller DrugSearchController
$ /Applications/XAMPP/xamppfiles/bin/php artisan make:controller TargetSearchController
```

Manually edit the controller files to include functions for extracting and storing information, as summarized in the above table.

All controller files can be retrieved here:

<https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/myproject/app/Http/Controllers>

## VIEW

Create files under resources/views/drugsearch and resources/views/targetsearch directories. The files should contain PHP and HTML codes that generate the presentation of data. View will use data from Controller to be presented to the user.

All PHP files under the view directory can be retrieved here:  
<https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/myproject/resources/views>

## ROUTES

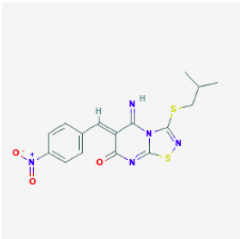
Routes can be defined in routes/web.php. '/' represents the homepage or welcome page (localhost/myproject/). At the beginning of the web.php, the route to Controller is invoked to link the view to the corresponding controller.

The PHP file containing all routes can be retrieved here:

<https://github.com/NurSyatila/XAMPP Laravel/blob/main/sampledata/myproject/routes/web.php>

# DrugDB

[HOME](#) [SEARCH](#) [APPROVED DRUGS](#) [DRUG TARGETS](#) [TARGET SEQUENCES](#)



**Sorafenib**  
*SmallMoleculeDrug*  
Drugbank: DB00398  
PubChem: 46505329  
PDB: BAX

**Indication:**  
Sorafenib is indicated for the treatment of unresectable hepatocellular carcinoma and advanced renal cell carcinoma.

**Protein targets for Sorafenib**

Show  entries Search:

Pharmacological action	Protein name	Gene name	GeneBank Protein ID	GeneBank Gene ID	Uniprot ID	Uniprot Title	PDBIDs	Species	View
Yes	Vascular endothelial growth factor receptor 3	FLT4	297050	X69878	P35916	VGFR3_HUMAN	<a href="#">View PDB</a>	Humans	<a href="#">View</a>
Yes	Vascular endothelial growth factor receptor 1	FLT1	31432	X51602	P17948	VGFR1_HUMAN	<a href="#">View PDB</a>	Humans	<a href="#">View</a>
Yes	RAF proto-oncogene serine/threonine-protein kinase	RAF1	35842	X03484	P04049	RAF1_HUMAN	<a href="#">View PDB</a>	Humans	<a href="#">View</a>

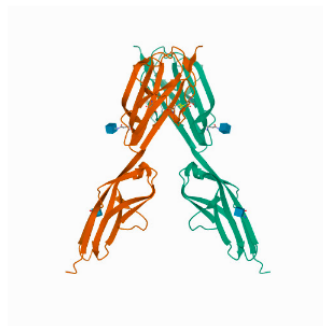
Showing 1 to 3 of 3 entries Previous  Next

(The View page from the web browser for individual drug)



# DrugDB

HOME SEARCH APPROVED DRUGS DRUG TARGETS TARGET SEQUENCES



Vascular endothelial growth factor receptor 3  
FLT4 | VEGFR3\_HUMAN  
Pharmacologically active

Uniprot: P35916

DrugBank: P35916

GeneBank (protein): 297050

GeneBank (gene): X69878

Go to PBD

## Drug molecules bound to Vascular endothelial growth factor receptor 3 (P35916)

Show 10 entries

Search:

DrugBank ID	Drug name	Drug Type	Indication	PubChem ID	HET ID	View
DB00398	Sorafenib	Small Molecule	Sorafenib is indicated for the treatment of unresectable hepatocellular carcinoma and advanced renal cell carcinoma.	46505329	BAX	<a href="#">View</a>
DB01268	Sunitinib	Small Molecule	For the treatment of advanced renal cell carcinoma as well as the treatment of gastrointestinal stromal tumor after disease progression on or intolerance to imatinib mesylate.	46507140	B49	<a href="#">View</a>
DB06589	Pazopanib	Small Molecule	Treatment of advanced renal cell cancer and advanced soft tissue sarcoma (in patients previously treated with chemotherapy)	175427074		<a href="#">View</a>
DB06626	Axitinib	Small Molecule	Used in kidney cell cancer and investigated for use/treatment in pancreatic and thyroid cancer.	347827779	AXI	<a href="#">View</a>

(The View page from the web browser for individual protein)

Now we have successfully created a website called DrugDB which provides queries and information on drugs, targets, and sequences.