

## Assignment 2 Report

All of the criteria items for this assessment has been successfully fulfilled except for *pagination*, *Image*, and *Recommendation*. The approach used to design this assignment product was to create migrations and seeder and model files for each table schema to check the functionality of the controller through routes in web.php as shown below.

```
264
265 Route::get('/follow', function () {
266     $follows = Follow::all();
267     //dd($votes);
268     foreach ($follows as $follow){
269         echo "User Name: {$follow->user->name} follows {$follow->follows->name} <br>";
270         echo "<br><br>";
271     }
272 });
273
274 // https://s2921450.elf.ict.griffith.edu.au/webAppDev/week9(20)/review/public/follow2
275 Route::get('/follow2', function () {
276     $users = User::all();
277     foreach($users as $user){
278         $id = $user->id;
279         $following = Follow::whereHas('user',function($query)use($id){
280             return $query->whereRaw('user_id = ?',array($id));
281         })->get();
282         echo "{$user->name} is following:<br>";
283         foreach($following as $follow){
284             echo "<ul>";
285             echo "<li>{$follow->follows->name}</li>";
286             echo "</ul>";
287         }
288     }
289 });
290
291 Route::get('/follow3', function () {
292     $users = User::all();
293     foreach($users as $user){
294         // $id = $user->id;
295         // $following = Follow::whereHas('user',function($query)use($id){
296         //     return $query->whereRaw('user_id = ?',array($id));
297         // })->get();
298         echo "{$user->name} is following:<br>";
299         foreach($user->following as $follow){
300             if(is_null($follow)){
301                 echo "<p>Not following anyone</p>";
302             }else{
303                 echo "<ul>";
304                 echo "<li>{$follow->name}</li>";
305                 echo "</ul>";
306             }
307         }
308     }
309 });
```

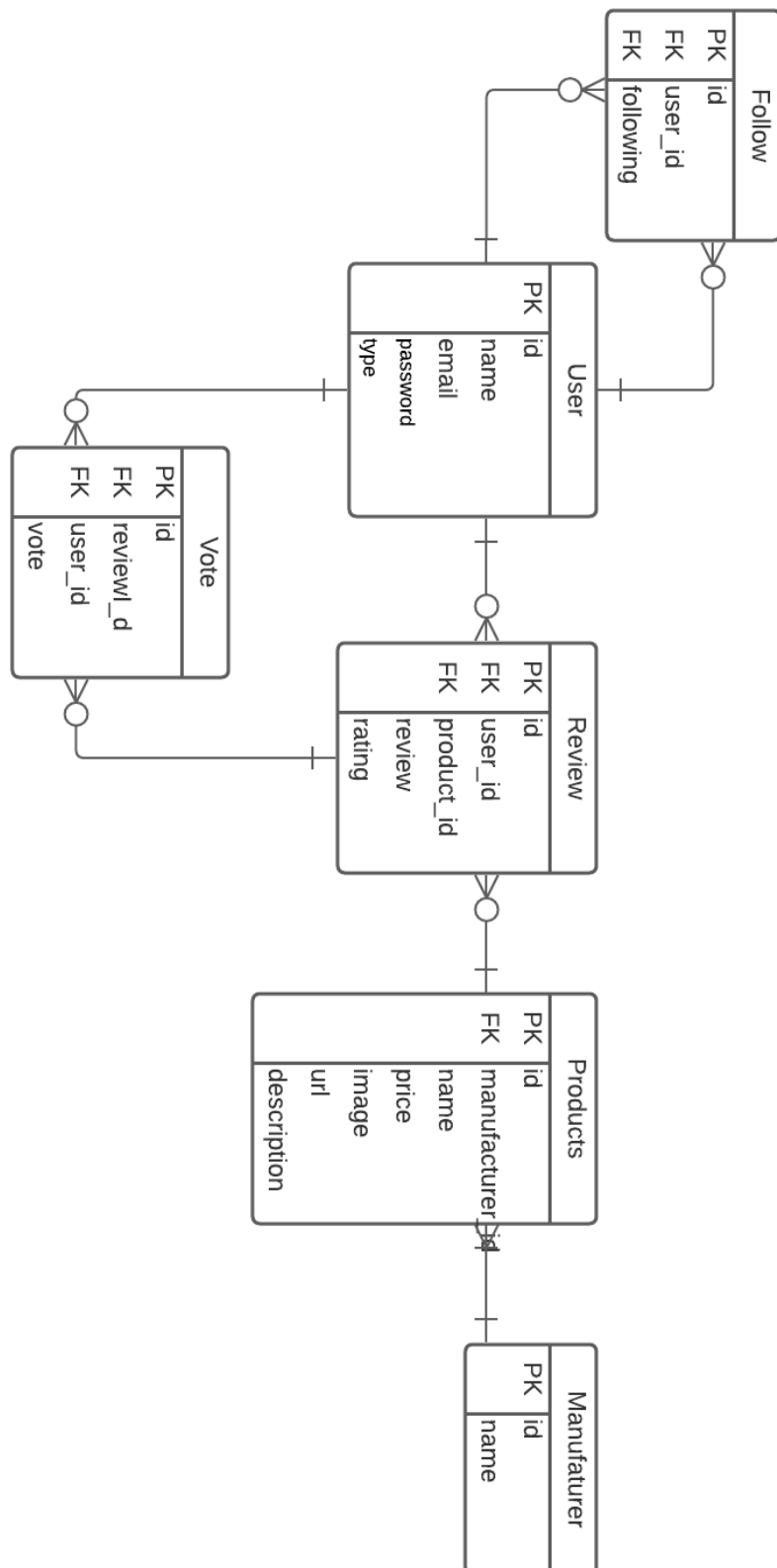
In order to effectively fulfil the criteria the design method chosen in this assignment was to create relational functions that use eloquent ORM capability rather than store related data in every table. The approach chosen in this assignment is illustrated by the code snippets below.

```
234
235 ✓ Route::get('/votes0', function () {
236     $votes = Vote::all();
237     //dd($votes);
238 ✓ foreach ($votes as $vote){
239     echo "Review ID: {{$vote->review->user->id}} <br>";
240     echo "Reviewer Name: {{$vote->review->user->name}} <br>";
241     echo "Comments: {{$vote->review->review}} <br>";
242
243     echo "Voter ID: {{$vote->review->user_id}} <br>";
244     echo "Voter Name: {{$vote->user->name}} <br>";
245 ✓ if($vote->vote == 1){
246     echo "Vote : 👍";
247 ✓ }else{
248     echo "Vote : 👎";
249     }
250     echo "<hr>";
251     echo "<br><br>";
252 }
253 };
```

```
159 public function show($id)
160
161 //
162 $product = Product::find($id);
163 $manufacturer = Product::find($id)->manufacturer;
164 $reviews = Review::whereRaw('product_id = ?',array($id))->get();
165 //dd($reviews);
166 $votes = [];
167 $results = [];
168 foreach($reviews as $review){
169     // $votes = Vote::whereRaw('review_id = ?',array($review->id))->get();
170     $votes = Vote::whereHas('review',function($query)use($id){
171         return $query->whereRaw('product_id = ?',array($id));
172     }->get());
173     // $results = DB::select("select votes.id,votes.vote,votes.created_at,votes.updated_at,votes.user_id,votes.review_id,products.id a
174     //dd($results);
175 //}
176 //dd($votes);
177 return view('products.show')->with('product',$product)->with('reviews',$reviews)->with('votes',$votes)->with('results',$results);
178 //return view('products.show')->with('product',product)->with('manufacturer',manufacturer);
179
180
```

Test first development using models and routes was used to develop the controller functions initially. Views were added to obtain instant feedback rather than relying on SQL commands on the terminal to test CRUD commands.

Before any code was written sequential pseudo-code design was made to avoid the program from crashing due the foreign key violation constraints. Algorithms were designed such that when deleting a product first the reviews and the votes (likes and dislikes) would be deleted first. Likewise, when deleting reviews the code ensured that the votes with the foreign keys referencing to the reviews are deleted first. This relationship is illustrated in the Entity relationship diagram below.



### Video Link

<https://web.microsoftstream.com/video/2abc1cf2-3627-4b93-9ccc-f1e5fdf0224f>