



ELECTRIC LAB AUTOMATION DOCUMENTATION 2025

PRESENTED TO

MISS KIRAN SHAKEEL

PRESENTED BY

S:NO	STD'S ID	STD'S NAME
1.	1547835	FIZZA ZULFIQAR
2.	1548125	RABEEA QASIM
3.	1544717	MADINA RASHID

BATCH CODE: PR2-2024-01B2

TABLE OF CONTENTS

- 1. INTRODUCTION**
- 2. PROBLEM DEFINATION**
- 3. CUSTOMER REQUIREMENT SPECIFICATION**
- 4. PROJECT PLANE**
- 5. E-R DIAGRAMS**
- 6. ALGORITHMS**
- 7. GUI STANDARDS DOCUMENTS**
- 8. INTERFACE DESIGN DOCUMENT**
- 9. TASK SHEET**
- 10. PROJECT REVIEW AND MONITORING REPORT**
- 11. UNIT TESTING CHECK LIST**
- 12. FINAL CHECK LIST**

INTRODUCTION

Electronic Lab Automation is a streamlined system designed to enhance the management and testing of electronic products. Built using Laravel and PHP with a MySQL database, it facilitates seamless product handling through three key user roles: Admin, User, and Tester. Admins can add and manage products, users can view and interact with them, and testers verify functionality with approval or disapproval options. The platform also includes a comment section for feedback and collaboration, ensuring an efficient and transparent workflow for electronics testing and validation.

PROBLEM DEFINITION

Traditional electronic product management and testing is slow, error-prone, and lacks transparency. Admins face challenges in managing products, testers struggle to provide structured feedback, and users have limited involvement in the testing process.

Electronic Lab Automation provides a digital platform where admins can manage products, testers can approve/disapprove and give feedback, and users can view and participate in discussions—ensuring efficiency, transparency, and accuracy in the workflow.

CUSTOMER REQUIREMENT SPECIFICATION

NON-FINANCIAL REQUIREMENTS

- *The system must store product details and*
- *testing data in a structured database.*
- *Unique 10-digit product ID and 12-digit test ID should be automatically generated.*
- *Advanced search options should be included.*
- *modular and sub-modular structure to manage different types of products and testing.*
- *Detailed remarks should be maintained covering test criteria and outcomes.*
- *The system should allow tracking of testing status.*
- *The name of the testing personnel should be recorded.*

CUSTOMER REQUIREMENT SPECIFICATION

FINANCIAL REQUIREMENTS

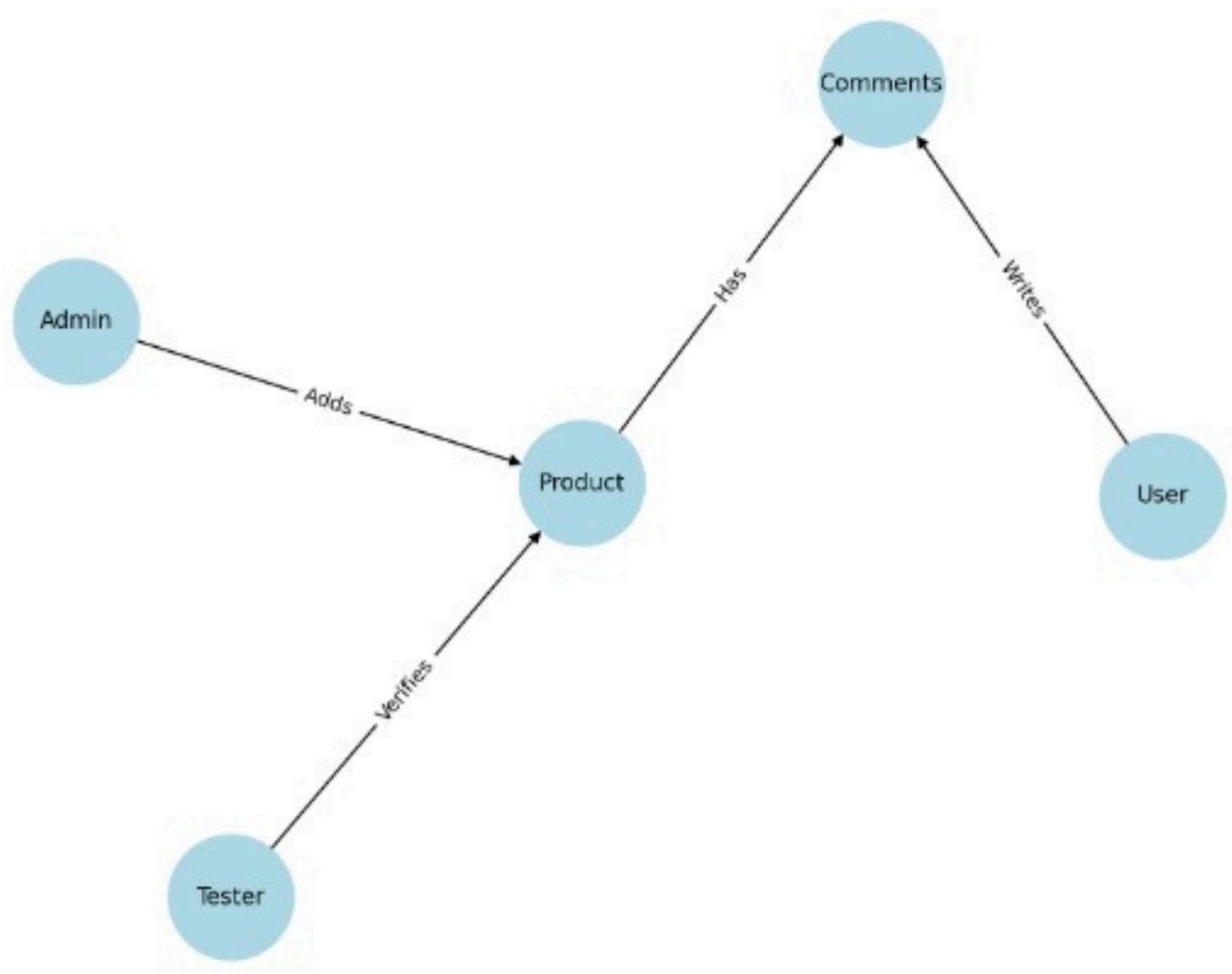
- *Failed products should be marked for remanufacturing.*
- *Successful products should proceed for further testing at CPRI for final approval*

PROJECT PLAN

PHASE	TASK	DURATION
PHASE 1	REQUIREMENT ANALYSIS	2 days
PHASE 2	DATABASE DESIGN	6 days
PHASE 3	FRONTEND & BACKEND DEVELOPMENT	2 weeks
PHASE 4	TESTING & DEBUGGING	4 days
PHASE 5	DEPLOYMENT	2 days

E-R DIAGRAMS

THE ENTITY-RELATIONSHIP DIAGRAM (ERD)
REPRESENTS THE LOGICAL STRUCTURE OF THE
DATABASE, INCLUDING ENTITIES SUCH AS
PRODUCTS, TESTING RECORDS, TESTERS
And test results.



ALGORITHMS

- **User/Admin/Tester logs in**
- **Selects a product**
- **Validate input (ensure it's not empty)**

GUI STANDARDS DOCUMENT

- **Dashboard:** Displays real-time testing statistics.
- **Product Management:** Interface for adding, modifying, and deleting product entries.
- **Testing Module:** Interface for recording test results and generating reports.
- **Search & Filters:** Advanced search functionality based on product ID, test ID, and test status.
- **Reports Section:** Generates reports for tested, failed, and approved products.

INTERFACE DESIGN DOCUMENT

- 1. Login Screen: Secure login for employees.**
- 2. Dashboard: Overview of testing progress and product details.**
- 3. Product Entry Form: Add new products and generate unique IDs.**
- 4. Testing Form: Record test details, results, and remarks.**
- 5. Search Page: Locate products using advanced search filters.**
- 6. Reports Page: View and export reports on product status.**

UNIT TESTING CHECKLIST

- 1. Product ID generation is working correctly.**
- 2. Test ID generation is accurate and unique.**
- 3. Search functionality returns expected results.**
- 4. Testing records stored and retrieved correctly.**
- 5. User authentication working securely.**

PROJECT REVIEW AND MONITORING REPORT

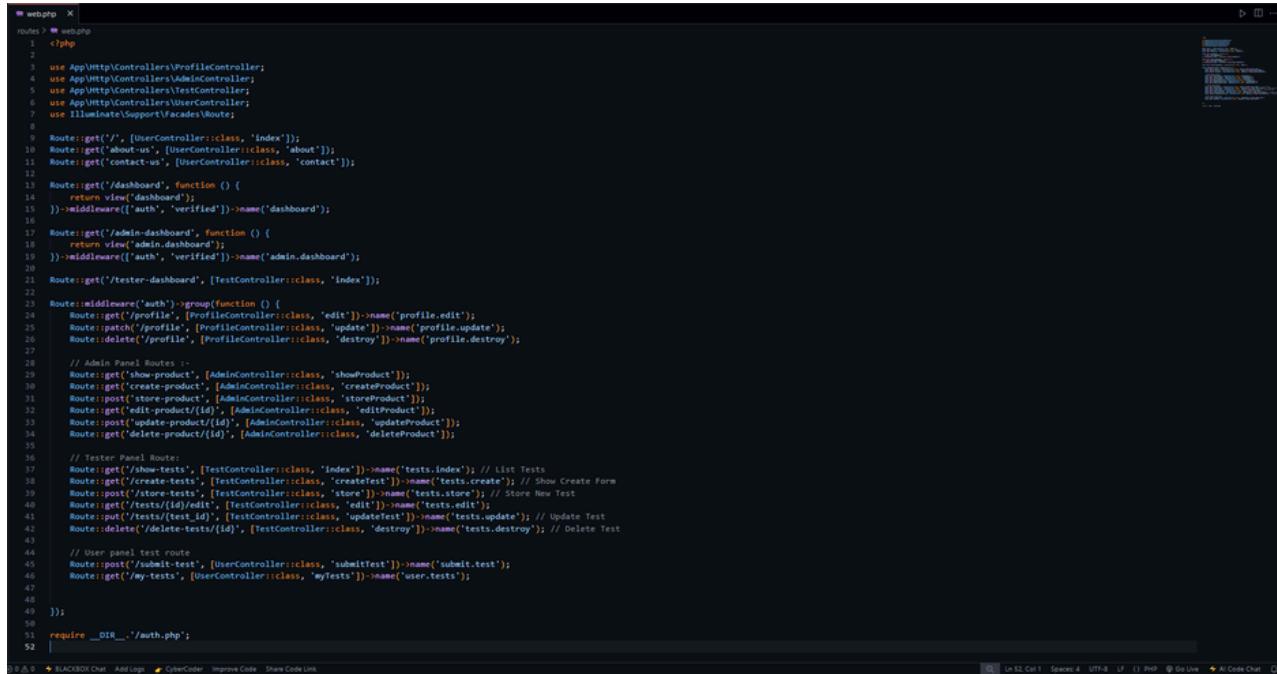
REGULAR PROJECT REVIEWS WILL BE CONDUCTED TO ENSURE:

- 1. Alignment with customer requirements.**
- 2. Timely completion of project phases.**
- 3. Functional and performance testing of modules.**

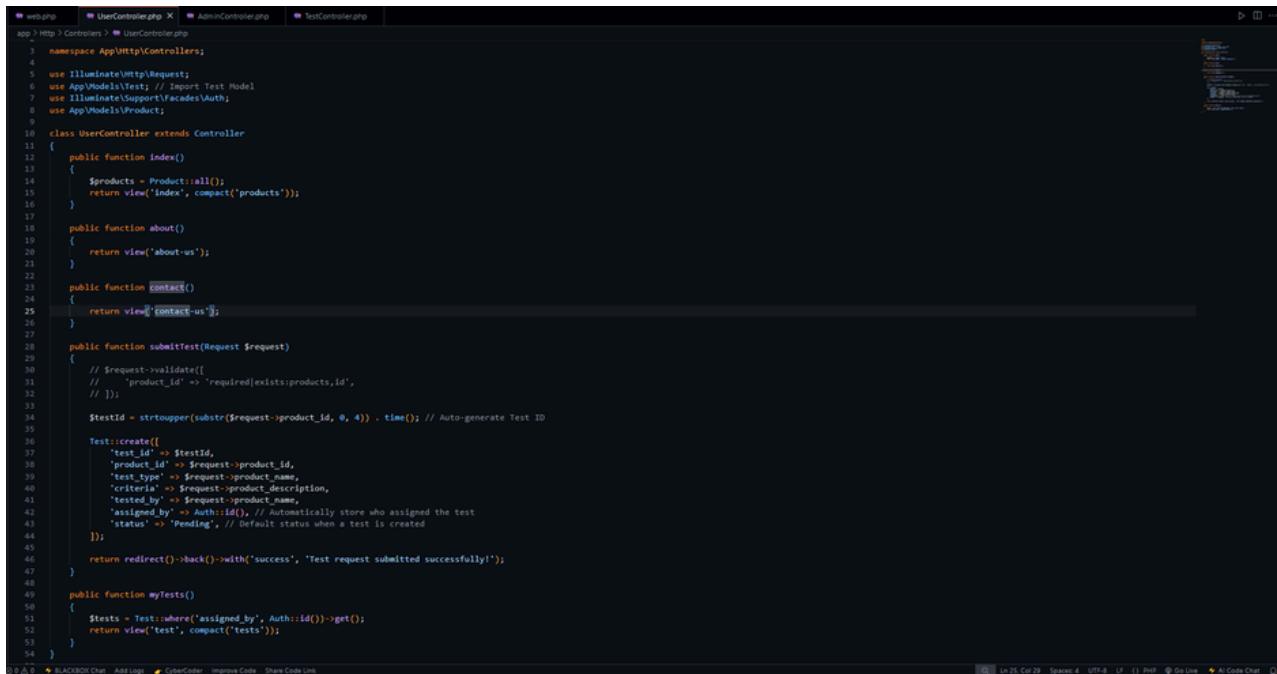
FINAL CHECKLIST

- ✓ *Dashboard for Admin, User, and Tester*
- ✓ *Admins can add and manage products.*
- ✓ *Testers can review, approve, or disapprove products.*
- ✓ *Users, admins, and testers can comment on products for feedback.*

CODE SCREEN SHOTS



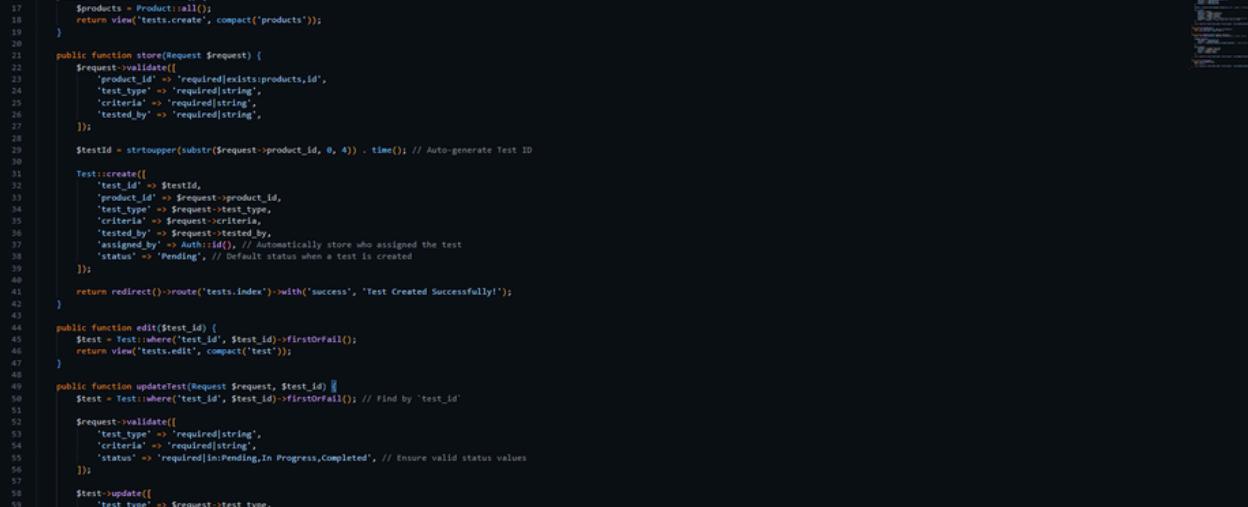
```
routes/web.php
1 <?php
2
3 use App\Http\Controllers\ProfileController;
4 use App\Http\Controllers\AdminController;
5 use App\Http\Controllers\TestController;
6 use Illuminate\Support\Facades\Route;
7 use Illuminate\Support\Acl;
8
9 Route::get('/', [UserController::class, 'index']);
10 Route::get('about-us', [UserController::class, 'about']);
11 Route::get('contact-us', [UserController::class, 'contact']);
12
13 Route::get('/dashboard', function () {
14     return view('dashboard');
15 })->middleware(['auth', 'verified'])->name('dashboard');
16
17 Route::get('/admin-dashboard', function () {
18     return view('admin.dashboard');
19 })->middleware(['auth', 'verified'])->name('admin.dashboard');
20
21 Route::get('/tester-dashboard', [TestController::class, 'index']);
22
23 Route::middleware('auth')->group(function () {
24     Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
25     Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
26     Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
27
28     // Admin Panel Routes
29     Route::get('show-product', [AdminController::class, 'showProduct'])->name('product.show');
30     Route::get('create-product', [AdminController::class, 'createProduct'])->name('product.create');
31     Route::post('store-product', [AdminController::class, 'storeProduct'])->name('product.store');
32     Route::get('edit-product/{id}', [AdminController::class, 'editProduct']);
33     Route::post('update-product/{id}', [AdminController::class, 'updateProduct']);
34     Route::get('delete-product/{id}', [AdminController::class, 'deleteProduct']);
35
36     // Tester Panel Routes
37     Route::get('/show-tests', [TestController::class, 'index'])->name('tests.index'); // List Tests
38     Route::get('/create-test', [TestController::class, 'createTest'])->name('tests.create'); // Show Create Form
39     Route::post('/store-test', [TestController::class, 'store'])->name('tests.store'); // Store New Test
40     Route::get('/tests/{id}/edit', [TestController::class, 'edit'])->name('tests.edit');
41     Route::put('/tests/{id}/edit', [TestController::class, 'updateTest'])->name('tests.update'); // Update Test
42     Route::delete('/delete-test/{id}', [TestController::class, 'destroy'])->name('tests.destroy'); // Delete Test
43
44     // User panel test route
45     Route::post('/submit-test', [UserController::class, 'submitTest'])->name('submit.test');
46     Route::get('/my-tests', [UserController::class, 'myTests'])->name('user.tests');
47
48 });
49
50 require __DIR__.'/_auth.php';
51
52 [
```



```
app/Http/Controllers > UserController.php
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\Test;
7 use Illuminate\Support\Facades\Auth;
8 use App\Models\Product;
9
10 class UserController extends Controller
11 {
12     public function index()
13     {
14         $products = Product::all();
15         return view('index', compact('products'));
16     }
17
18     public function about()
19     {
20         return view('about-us');
21     }
22
23     public function contact()
24     {
25         return view('contact-us');
26     }
27
28     public function submitTest(Request $request)
29     {
30         // $request->validate([
31         //     'product_id' => 'required|exists:products,id',
32         // ]);
33
34         $testId = strtoupper(substr($request->product_id, 0, 4)) . time(); // Auto-generate Test ID
35
36         Test::create([
37             'test_id' => $testId,
38             'product_id' => $request->product_id,
39             'test_type' => $request->product_name,
40             'criteria' => $request->product_description,
41             'tested_by' => $request->product_name,
42             'assigned_by' => Auth::id(), // Automatically store who assigned the test
43             'status' => 'Pending', // Default status when a test is created
44         ]);
45
46         return redirect()->back()->with('success', 'Test request submitted successfully!');
47     }
48
49     public function myTests()
50     {
51         $tests = Test::where('assigned_by', Auth::id())->get();
52         return view('test', compact('tests'));
53     }
54 }
```

CODE SCREEN SHOTS

```
 1 // 
 2 // 
 3 // 
 4 // 
 5 // 
 6 // 
 7 // 
 8 // 
 9 // 
10 // 
11 public function showProduct()
12 {
13     $products = Product::all();
14     return view('admin.products.index', compact('products'));
15 }
16 public function createProduct()
17 {
18     return view('admin.products.create');
19 }
20 public function storeProduct(Request $request)
21 {
22     $product = new Product();
23     $product->name = $request->input('pname');
24     $product->description = $request->input('pdescription');
25     $product->price = $request->input('pprice');
26     $product->quantity = $request->input('pqty');
27     if ($request->hasFile('pimage')) {
28         $img = time(). '-' . $request->pimage->getClientOriginalName();
29         $request->pimage->move(public_path('productsImages'), $img);
30         $product->image = $img;
31     }
32     $product->status = $request->input('pstatus');
33     $product->save();
34     return redirect('show-product');
35 }
36 public function editProduct($id)
37 {
38     $product = Product::find($id);
39     return view('admin.products.edit', compact('product'));
40 }
41 public function updateProduct(Request $request, $id)
42 {
43     $product = Product::find($id);
44     $product->name = $request->input('name');
45     $product->description = $request->input('pdescription');
46     $product->price = $request->input('pprice');
47     $product->quantity = $request->input('pqty');
48     if ($request->hasFile('pimage')) {
49         $img = time(). '-' . $request->pimage->getClientOriginalName();
50         $request->pimage->move(public_path('productsImages'), $img);
51         $product->image = $img;
52     }
53     $product->status = $request->input('pstatus');
54     $product->save();
55     return redirect('show-product');
56 }
57 public function deleteProduct($id)
58 {
59     Product::find($id)->delete();
60     return redirect('show-product');
61 }
```



The screenshot shows a PHP code editor with several tabs open. The active tab is 'TestController.php' under the 'Controllers' folder. The code implements a TestController class extending the Controller class. It includes methods for creating and storing tests, editing existing tests, and updating test details. Validation logic is provided using Request::validate() and Request::firstOrFail() methods.

```
app > http > Controllers > TestController.php
```

```
10 class TestController extends Controller {
11
12     public function createTest() {
13         $products = Product::all();
14         return view('tests.create', compact('products'));
15     }
16
17     public function store(Request $request) {
18         $request->validate([
19             'product_id' => 'required|exists:products,id',
20             'test_type' => 'required|string',
21             'criteria' => 'required|string',
22             'tested_by' => 'required|string',
23         ]);
24
25         $testId = strtoupper(substr($request->product_id, 0, 4)) . time(); // Auto-generate Test ID
26
27         Test::create([
28             'test_id' => $testId,
29             'product_id' => $request->product_id,
30             'test_type' => $request->test_type,
31             'criteria' => $request->criteria,
32             'tested_by' => $request->tested_by,
33             'assigned_by' => Auth::id(), // Automatically store who assigned the test
34             'status' => 'Pending', // Default status when a test is created
35         ]);
36
37         return redirect()->route('tests.index')->with('success', 'Test Created Successfully!');
38     }
39
40     public function edit($test_id) {
41         $test = Test::where('test_id', $test_id)->firstOrFail();
42         return view('tests.edit', compact('test'));
43     }
44
45     public function updateTest(Request $request, $test_id) {
46         $test = Test::where('test_id', $test_id)->firstOrFail(); // Find by 'test_id'
47
48         $request->validate([
49             'test_type' => 'required|string',
50             'criteria' => 'required|string',
51             'status' => 'required|in:Pending,In Progress,Completed', // Ensure valid status values
52         ]);
53
54         $test->update([
55             'test_type' => $request->test_type,
56             'criteria' => $request->criteria,
57             'status' => $request->status,
58             'result' => $request->result,
59         ]);
60
61         return redirect()->route('tests.index')->with('success', 'Test Updated Successfully!');
62     }
63 }
```

CODE SCREEN SHOTS

A screenshot of a code editor displaying the `User.php` file. The code is a PHP class definition for a User model, utilizing the Laravel framework's Eloquent ORM. It includes methods for mass assignment, hidden attributes, casting, and relationships. The code is well-formatted with syntax highlighting and code completion suggestions visible in the background.

```
1 <?php
2
3 namespace App\Models;
4
5 // use Illuminate\Contracts\Auth\MustVerifyEmail;
6 // use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8 use Illuminate\Notifications\Notifiable;
9
10 class User extends Authenticatable
11 {
12     /* @var string */
13     use HasFactory, Notifiable;
14
15     /**
16      * The attributes that are mass assignable.
17      *
18      * @var list<string>
19     */
20     protected $fillable = [
21         'name',
22         'email',
23         'password',
24     ];
25
26     /**
27      * The attributes that should be hidden for serialization.
28      *
29      * @var list<string>
30     */
31     protected $hidden = [
32         'password',
33         'remember_token',
34     ];
35
36     /**
37      * Get the attributes that should be cast.
38      *
39      * @return array<string, string>
40     */
41     protected function casts(): array
42     {
43         return [
44             'email_verified_at' => 'datetime',
45             'password' => 'hashed',
46         ];
47     }
48 }
```

A screenshot of a code editor displaying the `Test.php` file. This is a Model class for 'tests'. It defines the database table as 'tests', sets the primary key to 'test_id', and specifies that timestamps are tracked. The class includes mass assignment rules for fields like test_id, product_id, test_type, type, tested_by, assigned_by, result, and status. Relationships are defined with Product and User models. The code uses Eloquent's belongsTo and belongsTo methods respectively.

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Test extends Model
9 {
10     use HasFactory;
11
12     protected $table = 'tests';
13     protected $primaryKey = 'test_id'; // Set the correct primary key
14     public $timestamps = true;
15
16     // Allow mass assignment for these fields
17     protected $fillable = [
18         'test_id',
19         'product_id',
20         'test_type',
21         'type',
22         'tested_by',
23         'assigned_by', // NEW: Stores who assigned the test
24         'result', // NEW: Stores who assigned the test
25         'status', // NEW: Stores test status (Pending, In Progress, Completed)
26     ];
27
28     // Define relationship with Product
29     public function product() {
30         return $this->belongsTo(Product::class, 'product_id');
31     }
32
33     // Define relationship with User (Who Assigned the Test)
34     public function assignedBy() {
35         return $this->belongsTo(User::class, 'assigned_by');
36     }
37 }
```

CODE SCREEN SHOTS

A screenshot of a code editor displaying several PHP migration files. The files are part of a Laravel application's database migrations. The visible files include `0001_01_000000_create_users_table.php`, `0001_01_01_000000_create_products_table.php`, `2025_02_11_231647_create_products_table.php`, `2025_02_12_010529_create_techno_table.php`, and `2025_02_13_225955_update_techno_table.php`. The code is written in PHP using the Illuminate\Database\Migrations\Migration class and Schema facade.

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13     */
14    public function up(): void
15    {
16        Schema::create('users', function (Blueprint $table) {
17            $table->id();
18            $table->string('name');
19            $table->string('email')->unique();
20            $table->string('email_verified_at')->nullable();
21            $table->string('password');
22            $table->string('role')->default('user');
23            $table->rememberToken();
24            $table->timestamps();
25        });
26
27        Schema::create('password_reset_tokens', function (Blueprint $table) {
28            $table->string('email')->primary();
29            $table->string('token');
30            $table->timestamp('created_at')->nullable();
31        });
32
33        Schema::create('sessions', function (Blueprint $table) {
34            $table->string('id')->primary();
35            $table->foreignId('user_id')->nullable()->index();
36            $table->string('ip_address', 45)->nullable();
37            $table->text('user_agent')->nullable();
38            $table->longText('payload');
39            $table->integer('last_activity')->index();
40        });
41
42     /**
43      * Reverse the migrations.
44      *
45      * @return void
46     */
47     public function down(): void
48     {
49         Schema::dropIfExists('users');
50         Schema::dropIfExists('password_reset_tokens');
51         Schema::dropIfExists('sessions');
52     }
53 };
54 
```

A screenshot of a code editor displaying several PHP migration files. The files are part of a Laravel application's database migrations. The visible files include `0001_01_000000_create_users_table.php`, `0001_01_01_000000_create_products_table.php`, `2025_02_11_231647_create_products_table.php`, `2025_02_12_010529_create_techno_table.php`, and `2025_02_13_225955_update_techno_table.php`. The code is written in PHP using the Illuminate\Database\Migrations\Migration class and Schema facade.

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13     */
14    public function up(): void
15    {
16        Schema::create('products', function (Blueprint $table) {
17            $table->id();
18            $table->string('name');
19            $table->text('description');
20            $table->string('price');
21            $table->string('quantity');
22            $table->text('image')->nullable();
23            $table->string('status');
24            $table->timestamps();
25        });
26
27     /**
28      * Reverse the migrations.
29      *
30      * @return void
31     */
32     public function down(): void
33     {
34         Schema::dropIfExists('products');
35     }
36 };
37 
```

CODE SCREEN SHOTS

A screenshot of a code editor displaying a PHP migration file named `2025_02_13_010528_create_tests_table.php`. The file contains code for creating a table named 'tests' with columns for id (primary key), product_id (foreign key referencing 'products'), result (enum type), remarks (text), tested_by (text), and status (enum type). It also includes methods for up() and down() migrations.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('tests', function (Blueprint $table) {
            $table->string('test_id')->primary();
            $table->unsignedBigInteger('product_id'); // Match the data type of 'id' in products
            $table->foreign('product_id')->references('id')->onDelete('cascade'); // Reference 'id'
            $table->text('result');
            $table->enum('result', ['Pass', 'Fail'])->nullable();
            $table->text('remarks');
            $table->string('tested_by');
            $table->enum('status', ['Pending', 'In Progress', 'Completed'])->default('Pending');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('tests');
    }
};
```

A screenshot of a code editor displaying a PHP migration file named `2025_02_13_225955_update_tests_table.php`. The file contains code for updating the 'tests' table by removing the 'department_id' column and adding a new 'assigned_by' column (unsigned integer, nullable, after 'product_id'). It also adds a foreign key constraint for 'assigned_by' referencing 'users.id'. It includes methods for up() and down() migrations.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('tests', function (Blueprint $table) {
            // Remove unnecessary column
            $table->dropColumn('department_id');

            // Add new column
            $table->unsignedBigInteger('assigned_by')->nullable()->after('product_id');

            // Add foreign key constraint
            $table->foreign('assigned_by')->references('id')->onDelete('set null');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('tests', function (Blueprint $table) {
            // Re-add department_id if still needed
            $table->unsignedBigInteger('department_id')->nullable();
        });
    }
};
```

CODE SCREEN SHOTS

```
resources 2 views 3 admin 7 products > index.blade.php
```

```
2 <?section('content')>
3     <div class="wide_cont">
4         <div class="container-fluid">
5             <div class="row">
6                 <div class="col-md-12">
7                     <h2>All Product</h2>
8                 </div>
9             </div>
10            <div class="table_section padding_infor_info">
11                <div class="table-responsive-sm">
12                    <div class="d-flex justify-content-end">
13                        <a href="{{url('create-product')}}" class="btn btn-primary">Create</a>
14                    </div>
15                    <table class="table mt-2">
16                        <thead>
17                            <tr>
18                                <th scope="col">id</th>
19                                <th scope="col">Name</th>
20                                <th scope="col">Description</th>
21                                <th scope="col">Price</th>
22                                <th scope="col">Quantity</th>
23                                <th scope="col">Category</th>
24                                <th scope="col">Status</th>
25                                <th scope="col">Actions</th>
26                            </tr>
27                        </thead>
28                        <tbody>
29                            @foreach($products as $product)
30                            <tr>
31                                <td>{{ $product->id }}</td>
32                                <td>{{ $product->name }}</td>
33                                <td>{{ $product->description }}</td>
34                                <td>{{ $product->price }}</td>
35                                <td>{{ $product->quantity }}</td>
36                                <td>{{ $product->category }}</td>
37                                <td>{{ $product->status }}</td>
38                                <td>
39                                    <a href="{{url('edit-product/'.$product->id)}}" class="btn btn-warning">Edit</a>
40                                    <a href="{{url('delete-product/'.$product->id)}}" class="btn btn-danger">Delete</a>
41                                </td>
42                            </tr>
43                        </tbody>
44                    </table>
45                </div>
46            </div>
47        </div>
48    </div>
49    <!-- table section -->
50  </#section>
```

```
resources3:views3tests3 dashboardblade.php - Contracts dashboardblade.php - X indexblade.php - Index  
1 //extends('tests.master')  
2  
3 #section('content')  
4 <div class="container py-5">  
5   <h2>All Tests</h2>  
6   <a href="#" class="btn btn-primary my-3">Add New Test</a>  
7   <table class="table mt-3">  
8     <thead>  
9       <tr>  
10         <th>Test ID</th>  
11         <th>Product</th>  
12         <th>Test Type</th>  
13         <th>Assigned By</th>  
14         <th>Status</th>  
15         <th>Result</th>  
16         <th>Actions</th>  
17     </tr>  
18   </thead>  
19   <tbody>  
20     &foreach ($tests as $test)  
21       <tr>  
22         <td>{$test->test_id}</td>  
23         <td>{$test->product_name ?? 'N/A'}</td> <!-- Handle null product -->  
24         <td>{$test->test_type}</td>  
25         <td>{$test->assignedBy->name ?? 'Admin'}</td> <!-- Show Assigned By -->  
26         <td>  
27           &if($test->status == 'Pending')  
28             <span class="badge bg-warning">Pending</span>  
29           &elseif($test->status == 'In Progress')  
30             <span class="badge bg-info">In Progress</span>  
31           &elseif($test->status == 'Completed')  
32             <span class="badge bg-success">Completed</span>  
33           &else  
34             <span class="badge bg-secondary">Unknown</span>  
35           &endif  
36         </td>  
37         <td>{$test->result}</td>  
38         <td>  
39           <a href="{{ route('tests.edit', [$test->test_id]) }}" class="btn btn-warning btn-sm">Edit</a>  
40           <form action="{{ route('tests.destroy', [$test->test_id]) }}" method="POST" style="display:inline;">  
41             <csrf method="DELETE" />  
42             <button type="submit" class="btn btn-danger btn-sm" onclick="return confirm('Delete this test?')">Delete</button>  
43           </form>  
44         </td>  
45       </tr>  
46     &endforeach  
47   </tbody>  
48 </table>  
49 </div>  
50 #endsection  
51
```

CODE SCREEN SHOTS

```
resources 2 views 3 admin 7 products > index.blade.php index.blade.php index.php - views

2 @section('content')
3     <div class="wide-cont">
4         <div class="container-fluid">
5             <div class="row">
6                 <div class="col-md-12">
7                     <h2>All Product</h2>
8                 </div>
9             </div>
10            <div class="table-section padding_infor_info">
11                <div class="table-responsive-sm">
12                    <div class="d-flex justify-content-end">
13                        <a href="{{url('create-product')}}" class="btn btn-primary">Create</a>
14                    </div>
15                    <table class="table mt-2">
16                        <thead>
17                            <tr>
18                                <th scope="col">id</th>
19                                <th scope="col">Name</th>
20                                <th scope="col">Description</th>
21                                <th scope="col">Price</th>
22                                <th scope="col">Quantity</th>
23                                <th scope="col">Category</th>
24                                <th scope="col">Status</th>
25                                <th scope="col">Actions</th>
26                            </tr>
27                        </thead>
28                        <tbody>
29                            @foreach($products as $product)
30                                <tr>
31                                    <td>{{ $product->id }}</td>
32                                    <td>{{ $product->name }}</td>
33                                    <td>{{ $product->description }}</td>
34                                    <td>{{ $product->price }}</td>
35                                    <td>{{ $product->quantity }}</td>
36                                    <td>{{ $product->category }}</td>
37                                    <td>{{ $product->status }}</td>
38                                    <td>
39                                        <a href="{{url('edit-product/'.$product->id)}}" class="btn btn-warning">Edit</a>
40                                        <a href="{{url('delete-product/'.$product->id)}}" class="btn btn-danger">Delete</a>
41                                    </td>
42                                </tr>
43                            @endforeach
44                        </tbody>
45                    </table>
46                </div>
47            </div>
48        </div>
49    </div>
50    <!-- table section -->
51    @endsection
```

```
resources3:views3tests3 dashboardblade.php - Contracts dashboardblade.php - X indexblade.php - Index  
1 //extends('tests.master')  
2  
3 #section('content')  
4 <div class="container py-5">  
5     <h2>All Tests</h2>  
6     <a href="#" class="btn btn-primary my-3">Add New Test</a>  
7     <table class="table mt-3">  
8         <thead>  
9             <tr>  
10                <th>Test ID</th>  
11                <th>Product</th>  
12                <th>Test Type</th>  
13                <th>Assigned By</th>  
14                <th>Status</th>  
15                <th>Result</th>  
16                <th>Actions</th>  
17            </tr>  
18        </thead>  
19        <tbody>  
20            @foreach ($tests as $test)  
21                <tr>  
22                    <td>{{ $test->test_id }}</td>  
23                    <td>{{ $test->product->name ?? 'N/A' }}</td> <!-- Handle null product --&gt;<br/>24                    <td>{{ $test->test_type }}</td>  
25                    <td>{{ $test->assignedBy->name ?? 'Admin' }}</td> <!-- Show Assigned By --&gt;<br/>26                    <td>  
27                        @if($test->status == 'Pending')  
28                            <span class="badge bg-warning">Pending</span>  
29                        @elseif($test->status == 'In Progress')  
30                            <span class="badge bg-info">In Progress</span>  
31                        @elseif($test->status == 'Completed')  
32                            <span class="badge bg-success">Completed</span>  
33                        @else  
34                            <span class="badge bg-secondary">Unknown</span>  
35                        @endif  
36                    </td>  
37                    <td>{{ $test->result }}</td>  
38                    <td>  
39                        <a href="{{ route('tests.edit', [$test->test_id]) }}" class="btn btn-warning btn-sm">Edit</a>  
40                        <form action="{{ route('tests.destroy', [$test->test_id]) }}" method="POST" style="display:inline;">  
41                            @csrf @method('DELETE')  
42                            <button type="submit" class="btn btn-danger btn-sm" onclick="return confirm('Delete this test?')>Delete</button>  
43                        </form>  
44                    </td>  
45                </tr>  
46            @endforeach  
47        </tbody>  
48    </table>  
49 </div>  
50 #endsection  
51
```

DB SCREEN SHOTS

phpMyAdmin

Server: 127.0.0.1 > Database: lab_automation > Table: migrations

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Showing rows 0 - 5 (6 total, Query took 0.0004 seconds.)

SELECT * FROM `migrations`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

		id	migration	batch
<input type="checkbox"/>	Edit	1	0001_01_01_000000_create_users_table	1
<input type="checkbox"/>	Edit	2	0001_01_01_000001_create_cache_table	1
<input type="checkbox"/>	Edit	3	0001_01_01_000002_create_jobs_table	1
<input type="checkbox"/>	Edit	4	2025_02_11_231847_create_products_table	1
<input type="checkbox"/>	Edit	5	2025_02_12_010528_create_tests_table	1
<input type="checkbox"/>	Edit	6	2025_02_13_225955_update_tests_table	2

With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

[Print](#) [Copy to clipboard](#) [Export](#) [Display chart](#) [Create view](#)

[Bookmark this SQL query](#)

Console

The screenshot shows the phpMyAdmin interface for the 'migrations' table in the 'lab_automation' database. The table has columns: id, migration, and batch. The data consists of six rows, each representing a migration step. The first five rows are in batch 1, and the last one is in batch 2. The 'migration' column contains names like 'create_users_table', 'create_cache_table', etc.

Server: 127.0.0.1 > Database: lab_automation > Table: users

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

SELECT * FROM `users`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

		id	name	email	email_verified_at	password	role	remember_token	created_at	updated_at
<input type="checkbox"/>	Edit	1	admin	admin@gmail.com	NULL	\$2y\$12\$TNYX9KqQmp28U96xmd2dEOdYduGAocaAyAsIxFm2S3S...	admin	NULL	2025-02-12 01:28:49	2025-02-12 01:28:49
<input type="checkbox"/>	Edit	3	tester	tester@gmail.com	NULL	\$2y\$12\$YySmYgjW1QMSDM2uFIMZIB03CKzluouFB9gA0...	tester	NULL	2025-02-14 02:18:14	2025-02-14 02:18:14
<input type="checkbox"/>	Edit	4	user	user@gmail.com	NULL	\$2y\$12\$0cBOWSDQdghE26oAtnKKOZmJPq4u9Vpg9CzkvhJ...	user	NULL	2025-02-14 02:38:10	2025-02-14 02:38:10

With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

[Print](#) [Copy to clipboard](#) [Export](#) [Display chart](#) [Create view](#)

[Bookmark this SQL query](#)

Label: Let every user access this bookmark

[Bookmark this SQL query](#)

Console

The screenshot shows the phpMyAdmin interface for the 'users' table in the 'lab_automation' database. The table has columns: id, name, email, email_verified_at, password, role, remember_token, created_at, and updated_at. The data consists of three rows, each representing a user account. The roles are admin, tester, and user respectively.

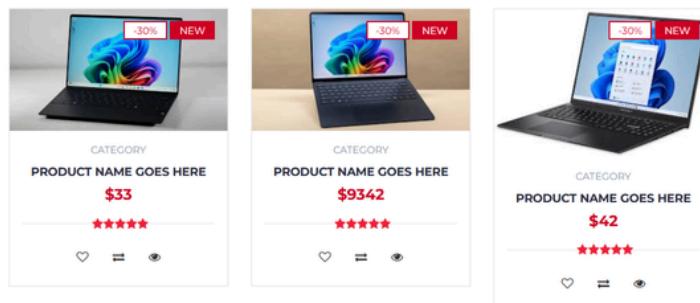
WEB SCREEN SHOTS

The screenshot shows the top navigation bar of the website. It includes contact information (+021-95-51-84, email@email.com, 1734 Stonecoal Road), a currency selector (\$ USD), and a log out link. Below the bar is a search bar with a placeholder "Search here" and a "Search" button. A dropdown menu labeled "All Categories" is visible.

The screenshot shows the main navigation menu at the top of the page. It includes links for Home, About US, My Tests, and Contact. The "Home" link is underlined, indicating it is the current page.



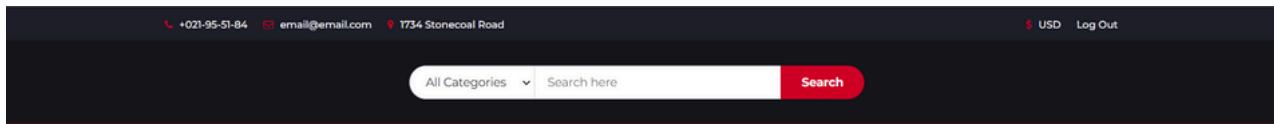
TOP TESTING



The screenshot shows the footer section of the website. It is divided into four main sections: "ABOUT US", "CATEGORIES", "INFORMATION", and "SERVICE". Each section contains links to various pages. Below these sections is a payment method section with icons for VISA, MasterCard, PayPal, American Express, and Discover. At the bottom, there is a copyright notice: "Copyright ©2025 All rights reserved | This template is made with ❤ by Colonia".

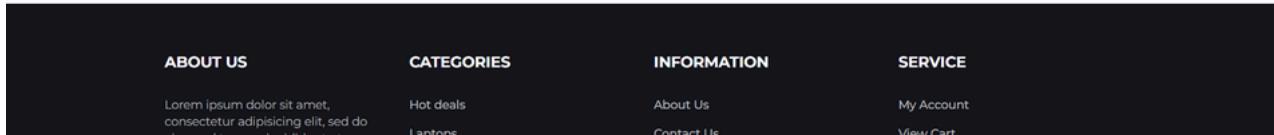
ABOUT US	CATEGORIES	INFORMATION	SERVICE
<p> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut.</p> <p>📍 1734 Stonecoal Road</p> <p>📞 +021-95-51-84</p> <p>✉️ email@email.com</p>	<p>Hot deals</p> <p>Laptops</p> <p>Smartphones</p> <p>Cameras</p> <p>Accessories</p>	<p>About Us</p> <p>Contact Us</p> <p>Privacy Policy</p> <p>Orders and Returns</p> <p>Terms & Conditions</p>	<p>My Account</p> <p>View Cart</p> <p>Wishlist</p> <p>Track My Order</p> <p>Help</p>

WEB SCREEN SHOTS

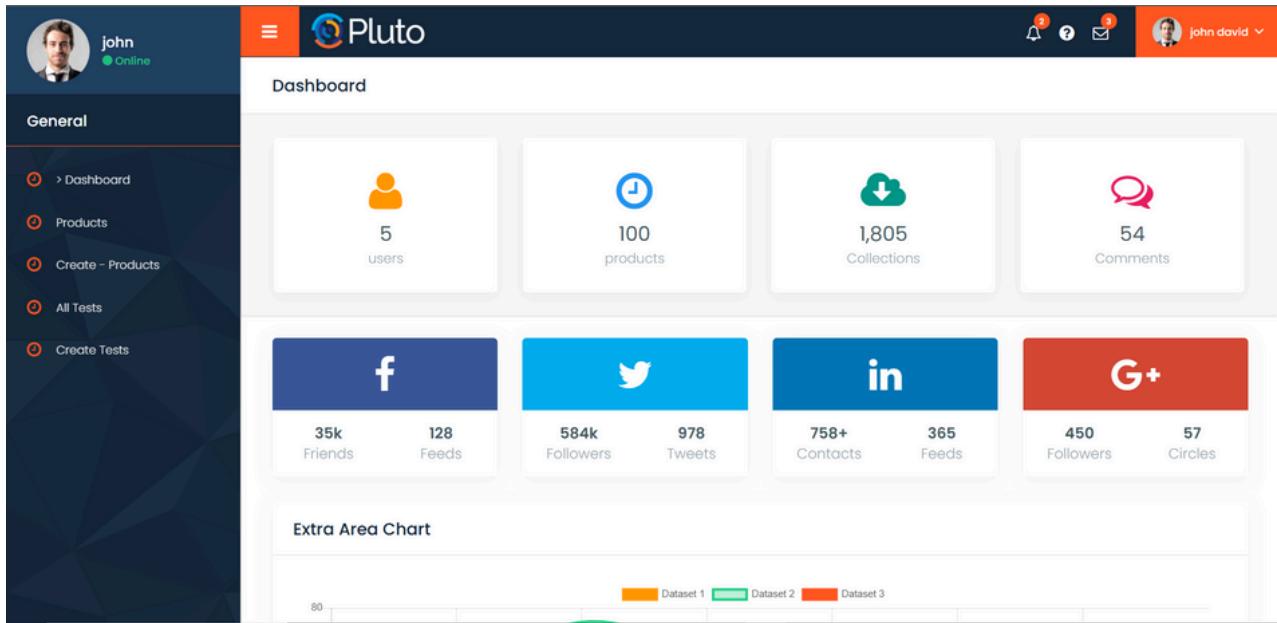


My Tests

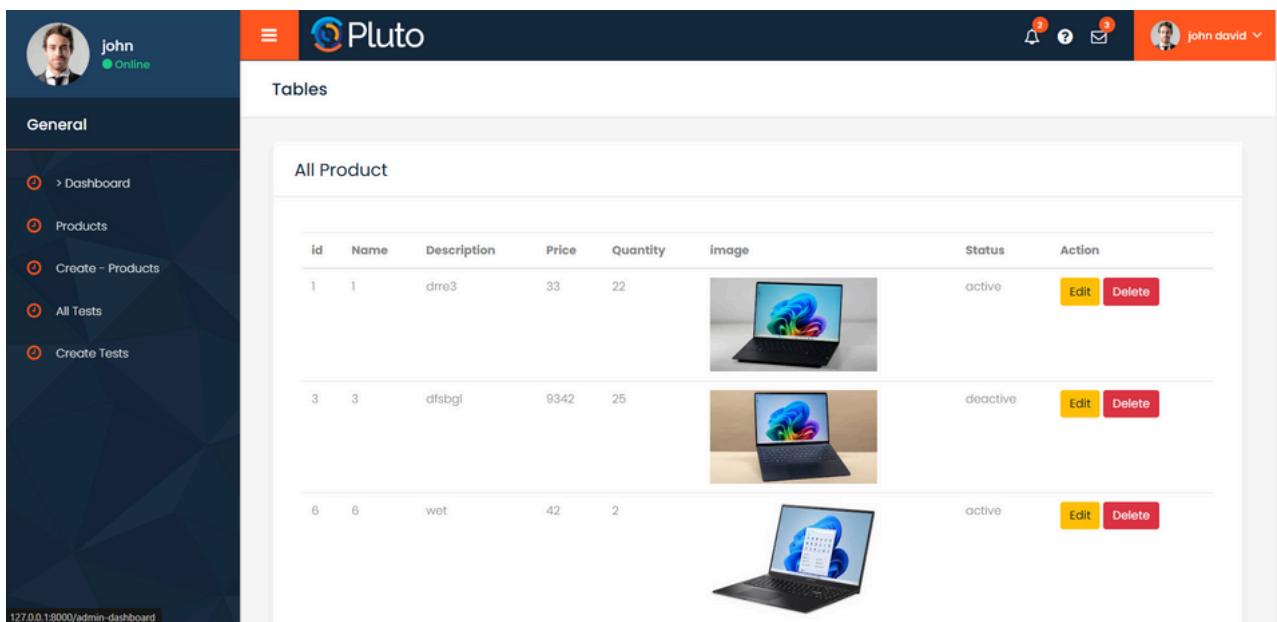
Product	Status	Result	Requested At
3	Completed	Pass	14 Feb 2025, 02:33
6	In Progress	Fail	14 Feb 2025, 02:26



ADMIN SCREEN SHOTS



The screenshot shows the Pluto Admin Dashboard. The top navigation bar includes a user profile for 'john' (online), a search icon, and notification icons for 2 messages, 1 question, and 3 emails. The dashboard header features the Pluto logo and the word 'Dashboard'. On the left, a sidebar titled 'General' lists navigation items: Dashboard, Products, Create - Products, All Tests, and Create Tests. The main content area displays four summary cards: 'users' (5), 'products' (100), 'Collections' (1,805), and 'Comments' (54). Below these are four social media summary cards for Facebook, Twitter, LinkedIn, and Google+ with their respective counts. A section for 'Extra Area Chart' is present but appears empty.



The screenshot shows the Pluto Admin Tables page. The top navigation bar is identical to the dashboard, showing 'john' (online) and notification counts. The header says 'Tables' and the sub-header is 'All Product'. The main content is a table listing products:

ID	Name	Description	Price	Quantity	Image	Status	Action
1	1	dirre3	33	22		active	<button>Edit</button> <button>Delete</button>
3	3	dfsbgf	9342	25		deactive	<button>Edit</button> <button>Delete</button>
6	6	wet	42	2		active	<button>Edit</button> <button>Delete</button>

ADMIN SCREEN SHOTS

The screenshot displays the Pluto Admin Dashboard. On the left, there's a sidebar titled 'General' with links: 'Dashboard', 'Products', 'Create - Products', 'All Tests', and 'Create Tests'. The main area is titled 'Add Product' and contains fields for 'Name', 'Description', 'Price', 'Quantity', a file upload section ('Choose File'), and a dropdown for 'Select Status'. A 'Submit' button is at the bottom. The top navigation bar shows the user 'john' is online, and the title 'Pluto'. The top right has notification icons and a user profile for 'john david'.

TESTER SCREEN SHOTS

The screenshot shows the 'All Tests' page of the Pluto application. The interface includes a dark sidebar on the left with a user profile for 'john' (online) and navigation links for 'General', 'All Tests', and 'Create Tests'. The main content area features a header 'All Tests' with a 'Add New Test' button. Below is a table listing three test entries:

Test ID	Product	Test Type	Assigned By	Status	Result	Actions
31739500425	3	sbl	admin	Completed	Pass	<button>Edit</button> <button>Delete</button>
61739499982	6	rwg	admin	In Progress	Fail	<button>Edit</button> <button>Delete</button>
61739500719	6	bracelete	user	Pending		<button>Edit</button> <button>Delete</button>

At the bottom, there is a copyright notice: 'Copyright © 2018 Designed by html.design. All rights reserved.' and 'Distributed By: ThemeWagon'.

The screenshot shows the 'Add New Test' page of the Pluto application. The interface includes a dark sidebar on the left with a user profile for 'john' (online) and navigation links for 'General', 'All Tests', and 'Create Tests'. The main content area features a header 'Add New Test' with a 'Select Product' dropdown set to '1'. Below are input fields for 'Test Type', 'Test Criteria', and 'Tested By'. A green 'Save Test' button is at the bottom. A copyright notice at the bottom states: 'Copyright © 2018 Designed by html.design. All rights reserved.' and 'Distributed By: ThemeWagon'.