As a Senior Backed Software developer , generate PHP code for the following , its frontend it located “./../frontend/” it’s a react   
  
# ADIS - Automated Drip Irrigation System

## Complete PHP Backend Implementation Guide

### Overview

This document provides comprehensive instructions for building a PHP backend service for the ADIS (Automated Drip Irrigation System) web application using XAMPP, MySQL, and PDO for database operations.

## Table of Contents

1. [System Requirements](#system-requirements)

2. [Database Structure](#database-structure)

3. [Project Structure](#project-structure)

4. [API Endpoints](#api-endpoints)

5. [PHP Implementation](#php-implementation)

6. [Security Considerations](#security-considerations)

7. [Testing](#testing)

## System Requirements

### XAMPP Setup

- XAMPP (Latest version with PHP 8.0+)

- MySQL/MariaDB

- Apache Server

- PHPMyAdmin for database management

### PHP Extensions Required

- PDO

- PDO\_MySQL

- JSON

- OpenSSL (for JWT tokens)

- cURL (for external API calls)

## Database Structure

### 1. Users Table

```sql

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

phone VARCHAR(20) UNIQUE NOT NULL,

village VARCHAR(255),

traditional\_authority VARCHAR(255),

district VARCHAR(255),

latitude DECIMAL(10, 8),

longitude DECIMAL(11, 8),

is\_admin BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

```

### 2. Products Table

```sql

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY,

product\_id VARCHAR(100) UNIQUE NOT NULL,

user\_id INT,

registration\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status ENUM('active', 'inactive', 'maintenance') DEFAULT 'active',

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL

);

```

### 3. OTP Verification Table

```sql

CREATE TABLE otp\_verifications (

id INT AUTO\_INCREMENT PRIMARY KEY,

phone VARCHAR(20) NOT NULL,

otp\_code VARCHAR(6) NOT NULL,

expires\_at TIMESTAMP NOT NULL,

is\_used BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

```

### 4. System Data Table

```sql

CREATE TABLE system\_data (

id INT AUTO\_INCREMENT PRIMARY KEY,

product\_id VARCHAR(100) NOT NULL,

soil\_moisture DECIMAL(5,2),

water\_usage\_today DECIMAL(8,2),

temperature DECIMAL(5,2),

humidity DECIMAL(5,2),

system\_status ENUM('active', 'inactive', 'error') DEFAULT 'active',

last\_watering TIMESTAMP,

next\_watering TIMESTAMP,

recorded\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (product\_id) REFERENCES products(product\_id) ON DELETE CASCADE

);

```

### 5. Crops Table

```sql

CREATE TABLE crops (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

crop\_name VARCHAR(100) NOT NULL,

planting\_date DATE NOT NULL,

growth\_stage ENUM('seedling', 'vegetative', 'flowering', 'fruiting', 'harvest') DEFAULT 'seedling',

watering\_schedule JSON,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

```

### 6. Support Tickets Table

```sql

CREATE TABLE support\_tickets (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

product\_id VARCHAR(100),

subject VARCHAR(255) NOT NULL,

description TEXT NOT NULL,

status ENUM('open', 'in\_progress', 'resolved', 'closed') DEFAULT 'open',

priority ENUM('low', 'medium', 'high', 'urgent') DEFAULT 'medium',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

```

### 7. Documents Table

```sql

CREATE TABLE documents (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

file\_name VARCHAR(255) NOT NULL,

file\_path VARCHAR(500) NOT NULL,

file\_size INT NOT NULL,

file\_type ENUM('pdf', 'video', 'image', 'manual') NOT NULL,

description TEXT,

is\_public BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

```

## Project Structure

```

backend/

├── config/

│ ├── database.php

│ ├── config.php

│ └── cors.php

├── controllers/

│ ├── AuthController.php

│ ├── UserController.php

│ ├── ProductController.php

│ ├── DashboardController.php

│ ├── SupportController.php

│ └── AdminController.php

├── models/

│ ├── User.php

│ ├── Product.php

│ ├── SystemData.php

│ ├── Crop.php

│ └── SupportTicket.php

├── middleware/

│ ├── AuthMiddleware.php

│ └── AdminMiddleware.php

├── utils/

│ ├── OTPService.php

│ ├── JWTHelper.php

│ ├── ResponseHelper.php

│ └── ValidationHelper.php

├── uploads/

│ ├── documents/

│ └── images/

├── api/

│ └── index.php

├── .htaccess

└── composer.json

```

## API Endpoints

### Authentication Endpoints

```

POST /api/auth/send-otp

POST /api/auth/verify-otp

POST /api/auth/logout

POST /api/auth/refresh-token

```

### User Management

```

POST /api/users/register

GET /api/users/profile

PUT /api/users/profile

GET /api/users/dashboard-data

```

### Product Management

```

POST /api/products/register

GET /api/products/my-products

GET /api/products/{product\_id}/data

PUT /api/products/{product\_id}/settings

```

### Crop Management

```

GET /api/crops

POST /api/crops

PUT /api/crops/{crop\_id}

DELETE /api/crops/{crop\_id}

```

### Support System

```

GET /api/support/tickets

POST /api/support/tickets

PUT /api/support/tickets/{ticket\_id}

GET /api/support/documents

```

### Admin Endpoints

```

GET /api/admin/users

GET /api/admin/products

GET /api/admin/analytics

PUT /api/admin/users/{user\_id}/status

```

## PHP Implementation

### 1. Database Configuration (config/database.php)

```php

<?php

class Database {

private $host = "localhost";

private $db\_name = "adis\_system";

private $username = "root";

private $password = "";

public $conn;

public function getConnection() {

$this->conn = null;

try {

$this->conn = new PDO(

"mysql:host=" . $this->host . ";dbname=" . $this->db\_name,

$this->username,

$this->password,

array(PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION)

);

} catch(PDOException $exception) {

echo "Connection error: " . $exception->getMessage();

}

return $this->conn;

}

}

?>

```

### 2. Main Configuration (config/config.php)

```php

<?php

// API Configuration

define('API\_VERSION', '1.0');

define('JWT\_SECRET', 'ADIS-secret-key');

define('JWT\_EXPIRY', 3600 \* 24); // 24 hours

// OTP Configuration

define('OTP\_EXPIRY', 300); // 5 minutes

define('OTP\_LENGTH', 6);

// File Upload Configuration

define('UPLOAD\_PATH', \_\_DIR\_\_ . '/../uploads/');

define('MAX\_FILE\_SIZE', 10 \* 1024 \* 1024); // 10MB

// SMS API Configuration (for OTP)

define('SMS\_API\_URL', 'your-sms-api-url');

define('SMS\_API\_KEY', 'your-sms-api-key');

// CORS Configuration

define('ALLOWED\_ORIGINS', ['http://localhost:3000', 'https://yourdomain.com']);

// Error Reporting

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

?>

```

### 3. CORS Handler (config/cors.php)

```php

<?php

function handleCORS() {

$origin = $\_SERVER['HTTP\_ORIGIN'] ?? '';

if (in\_array($origin, ALLOWED\_ORIGINS)) {

header("Access-Control-Allow-Origin: $origin");

}

header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");

header("Access-Control-Allow-Headers: Content-Type, Authorization, X-Requested-With");

header("Access-Control-Allow-Credentials: true");

if ($\_SERVER['REQUEST\_METHOD'] === 'OPTIONS') {

http\_response\_code(200);

exit();

}

}

?>

```

### 4. User Model (models/User.php)

```php

<?php

class User {

private $conn;

private $table\_name = "users";

public $id;

public $name;

public $phone;

public $village;

public $traditional\_authority;

public $district;

public $latitude;

public $longitude;

public $is\_admin;

public function \_\_construct($db) {

$this->conn = $db;

}

public function register() {

$query = "INSERT INTO " . $this->table\_name . "

SET name=:name, phone=:phone, village=:village,

traditional\_authority=:traditional\_authority,

district=:district, latitude=:latitude, longitude=:longitude";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":name", $this->name);

$stmt->bindParam(":phone", $this->phone);

$stmt->bindParam(":village", $this->village);

$stmt->bindParam(":traditional\_authority", $this->traditional\_authority);

$stmt->bindParam(":district", $this->district);

$stmt->bindParam(":latitude", $this->latitude);

$stmt->bindParam(":longitude", $this->longitude);

if($stmt->execute()) {

$this->id = $this->conn->lastInsertId();

return true;

}

return false;

}

public function findByPhone($phone) {

$query = "SELECT \* FROM " . $this->table\_name . " WHERE phone = :phone LIMIT 1";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":phone", $phone);

$stmt->execute();

if($stmt->rowCount() > 0) {

$row = $stmt->fetch(PDO::FETCH\_ASSOC);

$this->id = $row['id'];

$this->name = $row['name'];

$this->phone = $row['phone'];

$this->village = $row['village'];

$this->traditional\_authority = $row['traditional\_authority'];

$this->district = $row['district'];

$this->latitude = $row['latitude'];

$this->longitude = $row['longitude'];

$this->is\_admin = $row['is\_admin'];

return true;

}

return false;

}

public function updateProfile() {

$query = "UPDATE " . $this->table\_name . "

SET name=:name, village=:village,

traditional\_authority=:traditional\_authority,

district=:district, latitude=:latitude, longitude=:longitude

WHERE id=:id";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":name", $this->name);

$stmt->bindParam(":village", $this->village);

$stmt->bindParam(":traditional\_authority", $this->traditional\_authority);

$stmt->bindParam(":district", $this->district);

$stmt->bindParam(":latitude", $this->latitude);

$stmt->bindParam(":longitude", $this->longitude);

$stmt->bindParam(":id", $this->id);

return $stmt->execute();

}

}

?>

```

### 5. Authentication Controller (controllers/AuthController.php)

```php

<?php

require\_once '../config/database.php';

require\_once '../models/User.php';

require\_once '../utils/OTPService.php';

require\_once '../utils/JWTHelper.php';

require\_once '../utils/ResponseHelper.php';

class AuthController {

private $db;

private $user;

private $otpService;

public function \_\_construct() {

$database = new Database();

$this->db = $database->getConnection();

$this->user = new User($this->db);

$this->otpService = new OTPService($this->db);

}

public function sendOTP() {

$data = json\_decode(file\_get\_contents("php://input"));

if(empty($data->phone)) {

ResponseHelper::error("Phone number is required", 400);

return;

}

// Generate and send OTP

$otp = $this->otpService->generateOTP($data->phone);

if($this->otpService->sendSMS($data->phone, $otp)) {

ResponseHelper::success("OTP sent successfully");

} else {

ResponseHelper::error("Failed to send OTP", 500);

}

}

public function verifyOTP() {

$data = json\_decode(file\_get\_contents("php://input"));

if(empty($data->phone) || empty($data->otp)) {

ResponseHelper::error("Phone and OTP are required", 400);

return;

}

if($this->otpService->verifyOTP($data->phone, $data->otp)) {

// Check if user exists

if($this->user->findByPhone($data->phone)) {

$token = JWTHelper::encode([

'user\_id' => $this->user->id,

'phone' => $this->user->phone,

'is\_admin' => $this->user->is\_admin

]);

ResponseHelper::success("Login successful", [

'token' => $token,

'user' => [

'id' => $this->user->id,

'name' => $this->user->name,

'phone' => $this->user->phone,

'district' => $this->user->district,

'is\_admin' => $this->user->is\_admin

]

]);

} else {

ResponseHelper::error("User not found. Please register first.", 404);

}

} else {

ResponseHelper::error("Invalid or expired OTP", 400);

}

}

}

?>

```

### 6. OTP Service (utils/OTPService.php)

```php

<?php

class OTPService {

private $conn;

public function \_\_construct($db) {

$this->conn = $db;

}

public function generateOTP($phone) {

// Generate 6-digit OTP

$otp = sprintf("%06d", mt\_rand(100000, 999999));

// Store in database

$query = "INSERT INTO otp\_verifications

SET phone=:phone, otp\_code=:otp, expires\_at=:expires\_at";

$stmt = $this->conn->prepare($query);

$expires\_at = date('Y-m-d H:i:s', time() + OTP\_EXPIRY);

$stmt->bindParam(":phone", $phone);

$stmt->bindParam(":otp", $otp);

$stmt->bindParam(":expires\_at", $expires\_at);

if($stmt->execute()) {

return $otp;

}

return false;

}

public function verifyOTP($phone, $otp) {

$query = "SELECT \* FROM otp\_verifications

WHERE phone=:phone AND otp\_code=:otp

AND expires\_at > NOW() AND is\_used=0

ORDER BY created\_at DESC LIMIT 1";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":phone", $phone);

$stmt->bindParam(":otp", $otp);

$stmt->execute();

if($stmt->rowCount() > 0) {

// Mark OTP as used

$row = $stmt->fetch(PDO::FETCH\_ASSOC);

$update\_query = "UPDATE otp\_verifications SET is\_used=1 WHERE id=:id";

$update\_stmt = $this->conn->prepare($update\_query);

$update\_stmt->bindParam(":id", $row['id']);

$update\_stmt->execute();

return true;

}

return false;

}

public function sendSMS($phone, $otp) {

// Implement SMS sending logic here

// For development, you can log to file or return true

$message = "Your ADIS verification code is: $otp. Valid for 5 minutes.";

// Example using cURL for SMS API

/\*

$curl = curl\_init();

curl\_setopt\_array($curl, array(

CURLOPT\_URL => SMS\_API\_URL,

CURLOPT\_RETURNTRANSFER => true,

CURLOPT\_POST => true,

CURLOPT\_POSTFIELDS => json\_encode([

'to' => $phone,

'message' => $message,

'api\_key' => SMS\_API\_KEY

]),

CURLOPT\_HTTPHEADER => ['Content-Type: application/json']

));

$response = curl\_exec($curl);

curl\_close($curl);

\*/

// For development, just return true

error\_log("SMS OTP: $phone - $otp");

return true;

}

}

?>

```

### 7. JWT Helper (utils/JWTHelper.php)

```php

<?php

class JWTHelper {

public static function encode($payload) {

$header = json\_encode(['typ' => 'JWT', 'alg' => 'HS256']);

$payload['exp'] = time() + JWT\_EXPIRY;

$payload = json\_encode($payload);

$base64Header = str\_replace(['+', '/', '='], ['-', '\_', ''], base64\_encode($header));

$base64Payload = str\_replace(['+', '/', '='], ['-', '\_', ''], base64\_encode($payload));

$signature = hash\_hmac('sha256', $base64Header . "." . $base64Payload, JWT\_SECRET, true);

$base64Signature = str\_replace(['+', '/', '='], ['-', '\_', ''], base64\_encode($signature));

return $base64Header . "." . $base64Payload . "." . $base64Signature;

}

public static function decode($jwt) {

$tokenParts = explode('.', $jwt);

if(count($tokenParts) != 3) {

return false;

}

$header = base64\_decode(str\_replace(['-', '\_'], ['+', '/'], $tokenParts[0]));

$payload = base64\_decode(str\_replace(['-', '\_'], ['+', '/'], $tokenParts[1]));

$signatureProvided = $tokenParts[2];

$expiration = json\_decode($payload)->exp;

if($expiration < time()) {

return false;

}

$base64Header = str\_replace(['+', '/', '='], ['-', '\_', ''], base64\_encode($header));

$base64Payload = str\_replace(['+', '/', '='], ['-', '\_', ''], base64\_encode($payload));

$signature = hash\_hmac('sha256', $base64Header . "." . $base64Payload, JWT\_SECRET, true);

$base64Signature = str\_replace(['+', '/', '='], ['-', '\_', ''], base64\_encode($signature));

if($base64Signature === $signatureProvided) {

return json\_decode($payload, true);

}

return false;

}

}

?>

```

### 8. Response Helper (utils/ResponseHelper.php)

```php

<?php

class ResponseHelper {

public static function success($message = "Success", $data = null, $code = 200) {

http\_response\_code($code);

echo json\_encode([

'success' => true,

'message' => $message,

'data' => $data,

'timestamp' => date('Y-m-d H:i:s')

]);

exit;

}

public static function error($message = "Error", $code = 400, $errors = null) {

http\_response\_code($code);

echo json\_encode([

'success' => false,

'message' => $message,

'errors' => $errors,

'timestamp' => date('Y-m-d H:i:s')

]);

exit;

}

}

?>

```

### 9. Main API Router (api/index.php)

```php

<?php

require\_once '../config/config.php';

require\_once '../config/cors.php';

// Handle CORS

handleCORS();

// Set JSON header

header('Content-Type: application/json');

// Get request method and path

$method = $\_SERVER['REQUEST\_METHOD'];

$path = parse\_url($\_SERVER['REQUEST\_URI'], PHP\_URL\_PATH);

$path = str\_replace('/api', '', $path);

// Route handling

switch($path) {

case '/auth/send-otp':

if($method === 'POST') {

require\_once '../controllers/AuthController.php';

$controller = new AuthController();

$controller->sendOTP();

}

break;

case '/auth/verify-otp':

if($method === 'POST') {

require\_once '../controllers/AuthController.php';

$controller = new AuthController();

$controller->verifyOTP();

}

break;

case '/users/register':

if($method === 'POST') {

require\_once '../controllers/UserController.php';

$controller = new UserController();

$controller->register();

}

break;

case '/users/profile':

require\_once '../controllers/UserController.php';

$controller = new UserController();

if($method === 'GET') {

$controller->getProfile();

} elseif($method === 'PUT') {

$controller->updateProfile();

}

break;

case '/dashboard/data':

if($method === 'GET') {

require\_once '../controllers/DashboardController.php';

$controller = new DashboardController();

$controller->getDashboardData();

}

break;

default:

http\_response\_code(404);

echo json\_encode(['error' => 'Endpoint not found']);

break;

}

?>

```

### 10. .htaccess for URL Rewriting

```apache

RewriteEngine On

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule ^(.\*)$ api/index.php [QSA,L]

# Enable CORS for all requests

Header always set Access-Control-Allow-Origin "\*"

Header always set Access-Control-Allow-Methods "GET, POST, PUT, DELETE, OPTIONS"

Header always set Access-Control-Allow-Headers "Content-Type, Authorization, X-Requested-With"

```

## Security Considerations

### 1. Input Validation

- Validate all input data

- Use prepared statements for database queries

- Sanitize file uploads

- Implement rate limiting for OTP requests

### 2. Authentication & Authorization

- Use JWT tokens for session management

- Implement proper RBAC (Role-Based Access Control)

- Hash sensitive data

- Use HTTPS in production

### 3. Database Security

- Use environment variables for database credentials

- Implement proper error handling

- Use least privilege principle for database users

## Testing

### 1. API Testing with Postman

Create a Postman collection with all endpoints:

```json

{

"info": {

"name": "ADIS API",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"

},

"item": [

{

"name": "Auth",

"item": [

{

"name": "Send OTP",

"request": {

"method": "POST",

"header": [

{

"key": "Content-Type",

"value": "application/json"

}

],

"body": {

"mode": "raw",

"raw": "{\"phone\": \"+265888123456\"}"

},

"url": {

"raw": "http://localhost/adis-backend/api/auth/send-otp",

"protocol": "http",

"host": ["localhost"],

"path": ["adis-backend", "api", "auth", "send-otp"]

}

}

}

]

}

]

}

```

### 2. Unit Testing

Implement PHPUnit tests for critical functionality:

```php

<?php

use PHPUnit\Framework\TestCase;

class UserTest extends TestCase {

public function testUserRegistration() {

// Test user registration logic

$this->assertTrue(true);

}

public function testOTPGeneration() {

// Test OTP generation and validation

$this->assertTrue(true);

}

}

?>

```

## Deployment Instructions

### 1. XAMPP Setup

1. Install XAMPP

2. Start Apache and MySQL services

3. Create database 'adis\_system' in PHPMyAdmin

4. Import database schema

5. Place PHP files in htdocs/adis-backend/

### 2. Configuration

1. Update database credentials in config/database.php

2. Set proper JWT secret key

3. Configure SMS API credentials

4. Set appropriate file permissions

### 3. Production Considerations

1. Use environment variables for sensitive data

2. Enable HTTPS

3. Implement proper logging

4. Set up automated backups

5. Configure monitoring and alerts

## Additional Features to Implement

### 1. Real-time Data Integration

- WebSocket connections for live system data

- Integration with IoT sensors

- Automated irrigation scheduling

### 2. Analytics Dashboard

- Usage statistics

- Performance metrics

- Predictive analytics for crop management

### 3. Mobile App Support

- Push notifications

- Offline capability

- Location-based services