SQUARI

GUIDEBOOK

Panduan Sistem Monitoring Akuarium Berbasis IoT

DOSEN PEMBIMBING

Aditya Wicaksono S.Komp., M.Kom.
Lathifunnisa Fathonah M.T.

KETUA TIM

Nabil Rifqi Wijaya J0404231033

ANGGOTA TIM

Muhammad Sulthan Alfriansyah J0404231012
Daffa Rif'at Mahardika J0404231005

SEKOLAH VOKASI
IPB UNIVERSITY
2025

DAFTAR ISI

Pengenalan Produk 3

Fitur Produk 4

Tampilan Website 12

Penjelasan Teknologi 20

Alamat Domain 24

Kode Sumber 25

PENGENALAN PRODUK

Produk ini adalah Sistem Monitoring Real-Time Akuarium dan Lingkungan Sekitar dengan Fitur Auto-Feeding Berbasis Web. Sistem ini menggunakan NodeMCU ESP8266 yang tersambung ke sensor suhu, kelembapan, dan ketinggian air untuk mengumpulkan data secara berkelanjutan dan mengirimkannya ke Firebase Realtime Database. Antarmuka web dibangun dengan Laravel sehingga pengguna dapat memantau kondisi akuarium melalui browser.

Tujuan sistem ini adalah meningkatkan efisiensi dan akurasi dalam memantau parameter lingkungan akuarium serta mengotomasi proses pemberian pakan ikan. Sistem akan mengirimkan notifikasi jika parameter melewati batas aman sehingga pengguna dapat segera mengambil tindakan. Selain itu sistem ini juga berfungsi sebagai portofolio penerapan konsep Internet of Things dan pengembangan aplikasi web bagi praktikan.

Dalam penerapannya sensor BME280, DS18B20, dan HC-SR04 terhubung ke ESP8266 yang memproses dan mengirim data ke Firebase. Perintah pemberian pakan otomatis dieksekusi oleh servo motor sedangkan buzzer memberikan peringatan lokal saat kondisi kritis. Seluruh kontrol dan tampilan data ditampilkan pada dashboard web Laravel yang responsif sehingga cocok untuk budidaya ikan hias, akuarium penelitian, dan instalasi edukasi di laboratorium.

FITUR PRODUK

1. Pop Up Notifikasi

Fitur Pop Up Notifikasi berfungsi memberikan peringatan langsung kepada pengguna ketika nilai sensor melebihi atau berada di bawah ambang batas yang telah ditentukan. Notifikasi ini muncul otomatis di dashboard web, sehingga pengguna dapat segera mengetahui kondisi tidak normal, seperti suhu terlalu tinggi atau level air terlalu rendah.

2. Sinkronisasi Waktu Otomatis

Fitur Sinkronisasi Waktu Otomatis memastikan bahwa sistem selalu menggunakan waktu yang akurat dengan menyesuaikan jam secara otomatis berdasarkan zona waktu GMT melalui koneksi internet. Dengan menggunakan protokol NTP (Network Time Protocol), sistem dapat mencatat waktu setiap aktivitas dan data sensor secara konsisten tanpa perlu pengaturan manual.

FITUR PRODUK

3. Monitoring Sensor Real-Time

Data Sensor Terbaru

Sensor	Nilai	Satuan
Suhu Air (DS18B20)	25.0625	°C
Suhu Udara (BME280)	27.77	°C
Kelembaban	44.1377	%
Tekanan Udara	980.38885	hPa
Level Air	0	%

Terakhir update: -

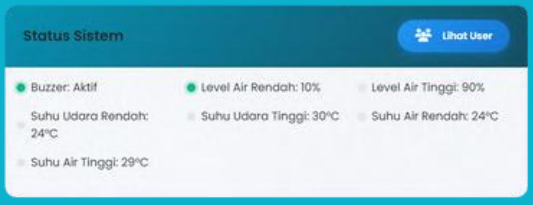
Fitur Monitoring Sensor Real-Time memungkinkan pengguna melihat data suhu udara, suhu air, dan ketinggian air secara langsung melalui dashboard web. Data dari sensor diperbarui secara otomatis tanpa perlu memuat ulang halaman, sehingga pengguna dapat memantau kondisi akuarium secara akurat dan responsif setiap saat.

FITUR PRODUK

4. Kontrol Pakan Jarak Jauh

Fitur kontrol pakan jarak jauh memungkinkan pengguna memberikan pakan ikan secara manual melalui tombol yang tersedia di dashboard web. Saat tombol ditekan, perintah dikirim ke mikrokontroler untuk menggerakkan servo motor yang mengatur pengeluaran pakan. Dengan fitur ini, pengguna dapat memberi pakan kapan saja tanpa harus berada di dekat akuarium.

5. Status Ambang Batas Sensor



Fitur ini berfungsi sebagai sistem peringatan otomatis berbasis ambang batas parameter lingkungan. Setiap nilai sensor seperti suhu udara, suhu air, dan ketinggian air memiliki batas minimum dan maksimum yang telah ditentukan, misalnya 10% dan 90% untuk level air. Jika data sensor melebihi atau kurang dari batas yang ditetapkan, sistem akan menampilkan notifikasi sebagai peringatan kepada pengguna.

FITUR PRODUK

6. Pengaturan Ambang Batas Sensor

Fitur Pengaturan Ambang Batas Sensor memungkinkan pengguna menentukan nilai minimum dan maksimum untuk setiap parameter lingkungan, seperti suhu air, suhu udara, dan ketinggian air. Nilai ambang batas ini digunakan sebagai acuan sistem untuk memberikan notifikasi jika kondisi melebihi batas normal.

7. Ekspor Data ke CSV

Fitur Ekspor Data ke CSV memungkinkan pengguna mengunduh seluruh data hasil pemantauan sensor dalam format file CSV. Data yang diekspor mencakup waktu pencatatan, suhu udara, suhu air, dan ketinggian air yang tersimpan di database.

FITUR PRODUK

8. Riwayat Data Sensor

Fitur Riwayat Data Sensor menyimpan seluruh data hasil pembacaan sensor secara berkala sehingga pengguna dapat melihat perkembangan kondisi akuarium dari waktu ke waktu. Data yang tersimpan mencakup suhu udara, suhu air, dan ketinggian air, lengkap dengan waktu pencatatannya.

9. Riwayat Data Notifikasi

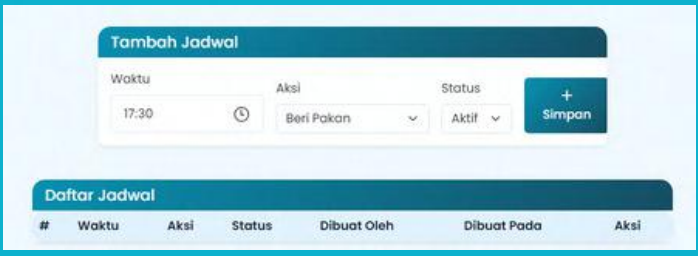
Fitur Riwayat Data Notifikasi menyimpan catatan semua notifikasi yang pernah muncul akibat parameter sensor melewati ambang batas. Setiap riwayat notifikasi mencakup informasi waktu kejadian, jenis peringatan, dan nilai sensor saat itu.

FITUR PRODUK

10. Riwayat Data Kontrol

Fitur Riwayat Data Kontrol mencatat setiap aktivitas pemberian pakan yang dilakukan melalui tombol di website. Data yang tersimpan mencakup waktu pelaksanaan, jenis aksi, status eksekusi, dan nama pengguna yang menjalankan perintah. Dengan fitur ini, pengguna dapat memantau jejak interaksi sistem secara transparan dan memastikan bahwa proses pemberian pakan berjalan sesuai perintah yang diberikan.

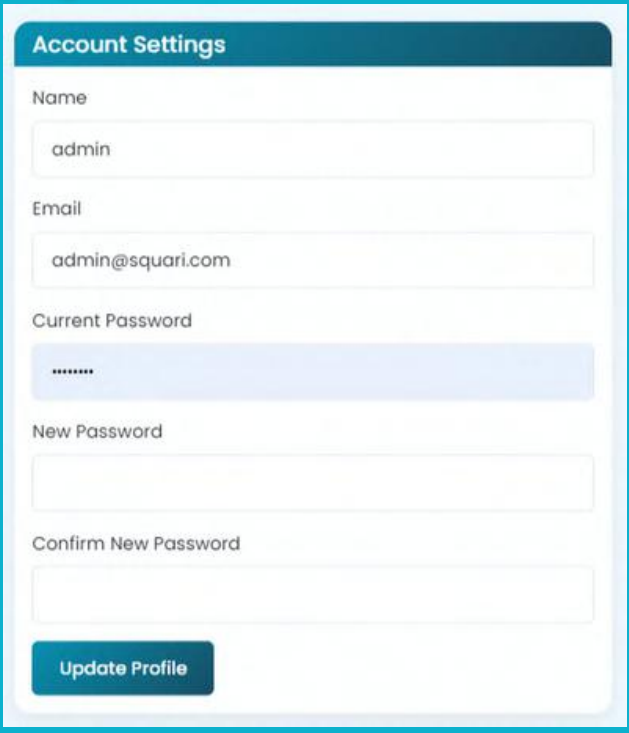
11. Kontrol Pakan Otomatis



Fitur Penjadwalan Pemberian Pakan memungkinkan pengguna mengatur waktu tertentu untuk menjalankan aksi seperti memberi pakan ikan secara otomatis. Pengguna dapat menentukan waktu, jenis aksi, status (aktif atau tidak), serta menyimpan jadwal melalui antarmuka web.

FITUR PRODUK

12. Pengaturan Ubah Profill



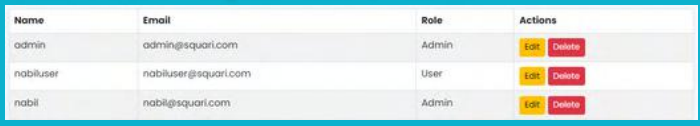
Fitur Pengaturan Ubah Profil memungkinkan pengguna memperbarui informasi akun mereka, termasuk mengganti kata sandi. Melalui antarmuka yang disediakan, pengguna dapat memasukkan kata sandi lama dan menetapkan kata sandi baru untuk menjaga keamanan akun. Fitur ini dirancang untuk memberikan kontrol penuh kepada pengguna terhadap data pribadinya dan meningkatkan perlindungan akses ke sistem.

FITUR PRODUK

13. Tambah User/Admin

Fitur Tambah User/Admin memungkinkan pengguna dengan hak akses tertentu untuk menambahkan akun baru ke dalam sistem, baik sebagai user biasa maupun sebagai admin. Melalui fitur ini, informasi seperti nama, email, peran, dan kata sandi dapat diatur saat pendaftaran akun baru.

14. Edit dan Hapus User/Admin



Fitur Edit dan Hapus User/Admin memungkinkan pengelola sistem untuk memperbarui informasi akun pengguna yang sudah terdaftar atau menghapusnya dari sistem. Melalui fitur ini, data seperti nama, email, peran, dan status akun dapat disesuaikan jika terjadi perubahan.

TAMPILAN WEBSITE

User Interface

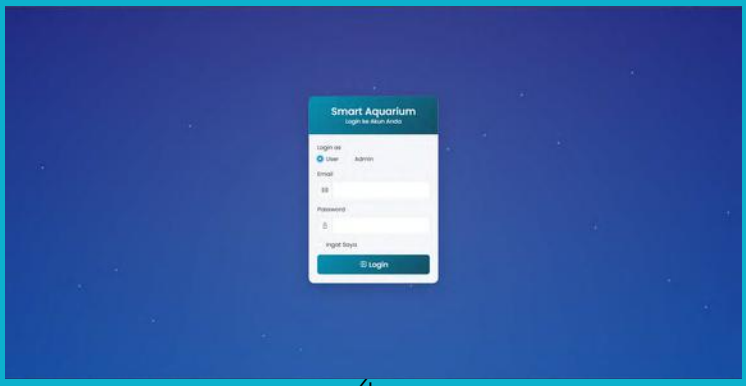
Website Smart Aquarium memiliki beberapa halaman utama:

HomePage

Tampilan awal website menampilkan header dengan logo Smart Aquarium, menu navigasi, serta latar belakang akuarium animatif dengan air, gelembung, dan ikan bergerak. Tersedia tombol Login/Register, deskripsi singkat produk, fitur unggulan, serta menu kontak dan bantuan.

TAMPILAN WEBSITE

Halaman Login



4

Halaman login adalah pintu masuk ke sistem Smart Aquarium. Pengguna memilih peran (User/Admin) lalu memasukkan email dan password untuk mengakses dashboard.

Elemen Tampilan:

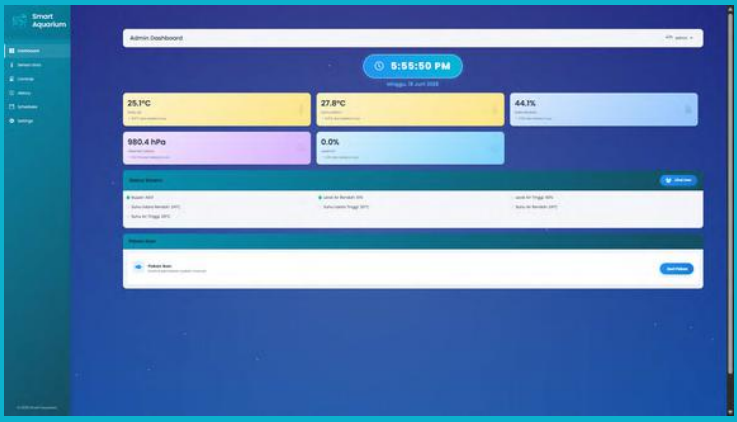
- Form Login:
  - Email: dengan validasi format & pesan error otomatis
  - Password: dilengkapi toggle show/hide
  - Ingat Saya: menyimpan sesi login
  - Tombol Login: untuk autentikasi

Fitur Keamanan:

- Validasi input real-time
- Proteksi brute force
- Manajemen sesi dengan token "Remember Me"

TAMPILAN WEBSITE

Dashboard



Dashboard merupakan pusat kendali utama yang menampilkan ringkasan kondisi akuarium secara real-time. Pengguna dapat memantau suhu air dan udara, kelembaban, tekanan, serta level air langsung dari berbagai sensor. Setiap data dilengkapi dengan indikator tren dan batas normal untuk memudahkan pemantauan.

Selain data sensor, dashboard juga menampilkan status sistem seperti kondisi buzzer, mode operasi, dan peringatan level air. Fitur tambahan mencakup jam digital, status koneksi (Firebase dan perangkat), serta informasi profil pengguna seperti nama dan foto dengan menu akses cepat.

TAMPILAN WEBSITE

Sensor Data

Halaman Sensor Data menampilkan informasi terbaru dari berbagai sensor yang terpasang pada akuarium. Di bagian atas, pengguna dapat melihat waktu saat ini serta tabel berisi data real-time seperti suhu air, suhu udara, kelembaban, tekanan udara, dan level air. Seluruh data ditampilkan beserta satuannya dan diperbarui secara berkala. Di bawahnya terdapat grafik visual untuk masing-masing sensor yang menunjukkan tren perubahan data dari waktu ke waktu. Tampilan ini memudahkan pengguna dalam memantau kondisi akuarium secara menyeluruh dan cepat.

TAMPILAN WEBSITE

Controls

Halaman Kontrol kendali sistem:

- Beri Pakan: Memberikan pakan secara manual ke akuarium.
- Status Buzzer: Menampilkan apakah buzzer aktif, digunakan untuk peringatan otomatis.

Kontrol Rentang Sensor:

- Suhu Udara & Suhu Air (Min/Max): Menentukan batas aman suhu lingkungan dan air.
- Level Air (Low/High): Mengatur batas bawah dan atas level air dalam tangki.
- Simpan Rentang: Menyimpan pengaturan sensor yang telah disesuaikan.

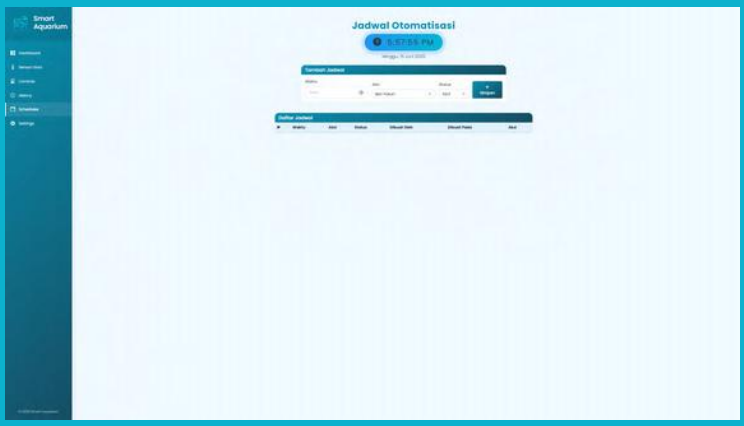
TAMPILAN WEBSITE

History

Halaman History menampilkan catatan lengkap dari aktivitas dan data sensor yang terekam oleh sistem Smart Aquarium. Pengguna dapat melihat log seperti suhu, kelembapan, tekanan, hingga level air dari waktu ke waktu dalam bentuk tabel yang terstruktur. Selain itu, riwayat kontrol dan notifikasi juga tersedia, memungkinkan pemantauan terhadap aktivitas perangkat maupun pemberitahuan sistem. Fitur pencarian, filter berdasarkan waktu atau kategori, serta ekspor data ke format CSV disediakan untuk memudahkan analisis dan dokumentasi.

TAMPILAN WEBSITE

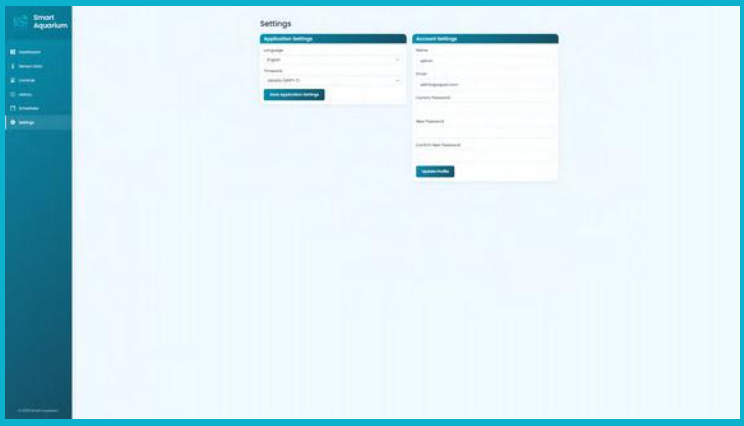
Schedules



Halaman Schedule Otomatisasi memungkinkan pengguna mengatur berbagai aktivitas otomatis dalam sistem Smart Aquarium. Di sini, pengguna dapat menambahkan jadwal seperti pemberian pakan secara berkala dengan memilih waktu, jenis aksi, dan status aktif atau tidak. Semua jadwal yang telah dibuat ditampilkan dalam tabel yang rapi, mencakup informasi waktu, aksi yang dijalankan, status, serta data pembuat dan waktu pembuatan. Tampilan yang sederhana ini memudahkan pengguna untuk mengelola, mengedit, atau menghapus jadwal sesuai kebutuhan, sehingga operasional akuarium dapat berjalan otomatis dan efisien.

TAMPILAN WEBSITE

Settings



Pada halaman Setting, pengguna dapat menyesuaikan preferensi aplikasi sesuai kebutuhan.

Application Settings:

- Pilihan bahasa antarmuka
- Pengaturan zona waktu sistem
- Tombol simpan pengaturan aplikasi

Account Settings:

- Ubah nama pengguna
- Ubah alamat email
- Ganti kata sandi (masukkan sandi lama dan baru)
- Konfirmasi kata sandi baru
- Tombol update profil

Penjelasan Teknologi

Arsitektur Sistem

Sistem Squari dirancang menggunakan pendekatan Internet of Things (IoT) yang menggabungkan perangkat keras (sensor, mikrokontroler) dengan perangkat lunak (cloud database dan web interface). Tujuan dari arsitektur ini adalah untuk memungkinkan pemantauan dan pengendalian akuarium secara real-time dari mana saja.

Komponen utama dalam arsitektur sistem :

- **Sensor**, Bertugas mengukur data fisik seperti suhu dan kelembapan udara, suhu air, dan ketinggian air, kemudian data sensor dikirim ke mikrokontroler untuk diproses.
- **NodeMCU ESP8266**, Merupakan mikrokontroler berbasis Wi-Fi yang berfungsi sebagai otak sistem, bertugas dengan mengumpulkan data dari sensor, mengirimkannya ke Firebase Realtime Database, dan menerima perintah dari pengguna (misalnya untuk pemberian pakan).
- **Firebase Realtime Database**, Digunakan sebagai tempat penyimpanan data sensor dan status sistem, serta Mampu memperbarui data secara real-time dan sinkron ke semua klien yang terhubung.
- **Website Laravel**, Digunakan sebagai antarmuka pengguna untuk menampilkan data, grafik histori, notifikasi, dan fitur kontrol (auto-feeding, pengaturan ambang batas, dsb).

Penjelasan Teknologi

Arsitektur Sistem

- **Aktuator (Servo Motor dan Buzzer)**, Servo motor berfungsi untuk membuka wadah pakan sesuai jadwal atau perintah pengguna, sedangkan buzzer memberikan peringatan saat suhu atau tinggi air melebihi ambang batas. Keduanya berperan penting dalam merespons kondisi lingkungan akuarium.

Komponen Hardware

Sistem Squari dibangun menggunakan sejumlah komponen elektronik utama yang bekerja secara terintegrasi untuk menjalankan fungsi pemantauan kondisi akuarium dan pemberian pakan otomatis. Berikut penjelasan masing-masing komponennya.

- **NodeMCU ESP8266**, Berperan sebagai pusat kendali seluruh sistem. Mikrokontroler ini sudah dilengkapi modul Wi-Fi internal, sehingga mampu terhubung ke internet tanpa tambahan perangkat. ESP8266 membaca data dari semua sensor yang terpasang, lalu mengirimkannya secara real-time ke Firebase Realtime Database. Selain itu, perangkat ini juga menerima perintah dari website untuk mengendalikan aktuator seperti servo dan buzzer, serta merespons input dari tombol manual.
- **Sensor DS18B20**, Sensor ini digunakan untuk mengukur suhu air dalam akuarium.

Penjelasan Teknologi

Komponen Hardware

- **Sensor BME280**, Dalam sistem Squari, sensor ini digunakan untuk mendapatkan informasi suhu dan kelembapan udara di sekitar akuarium. Data dari BME280 membantu pengguna memahami kondisi lingkungan secara menyeluruh.
- **Sensor Ultrasonik HC-SR04**, Sensor ini digunakan untuk mengukur ketinggian permukaan air dalam akuarium. Sensor ini bekerja dengan memancarkan gelombang suara ultrasonik yang dipantulkan oleh permukaan air, kemudian menghitung waktu pantulan untuk menentukan jarak. Jika ketinggian air berada di bawah ambang batas yang ditentukan, NodeMCU akan memicu buzzer dan mengirimkan peringatan ke website sebagai bentuk notifikasi kepada pengguna.
- **Servo Motor**, Aktuator ini digunakan untuk membuka dan menutup penutup tempat pakan ikan. Motor ini dikendalikan oleh NodeMCU berdasarkan waktu yang telah dijadwalkan pengguna melalui website atau melalui perintah manual. Mekanisme ini memungkinkan pemberian pakan ikan dilakukan secara otomatis dan konsisten, tanpa perlu intervensi manual setiap saat.
- **Buzzer dan Tombol Manual**, Buzzer berfungsi sebagai alat peringatan yang berbunyi ketika terjadi kondisi tidak normal, seperti suhu terlalu tinggi atau air terlalu rendah. Selain itu, Squari juga dilengkapi tombol manual yang memungkinkan pengguna memberikan pakan secara langsung dengan menekan tombol fisik pada perangkat.

Penjelasan Teknologi

Platform dan Framework

Untuk menghubungkan perangkat Squari dengan pengguna secara efisien, Squari memanfaatkan beberapa platform dan framework modern yang mendukung kestabilan sistem, kemudahan pengembangan, dan antarmuka pengguna yang responsif.

- **Firebase Realtime Database**, Digunakan sebagai basis penyimpanan data utama dalam sistem Squari. Platform ini memungkinkan data yang dikirim dari NodeMCU (seperti suhu, kelembapan, ketinggian air, dan status aktuator) langsung tersimpan dan tersinkron secara real-time dengan antarmuka web pengguna. Firebase juga mendukung fitur notifikasi berbasis event, sehingga ketika terjadi perubahan nilai yang signifikan (misalnya suhu melebihi ambang batas), sistem dapat memberikan peringatan secara instan.
- **Laravel**, Digunakan untuk membangun sistem backend dan frontend website Squari. Versi terbaru Laravel menawarkan kestabilan, keamanan, dan kemudahan pengembangan.

Dengan kombinasi platform dan framework ini, Squari dapat memberikan pengalaman pengguna yang optimal, stabilitas sistem yang baik, dan fleksibilitas tinggi untuk pengembangan lebih lanjut.

ALAMAT DOMAIN

Akses Website Squari
dengan cepat

https://smart-aquarium.rf.gd/

Kode Sumber

routes/web.php

Penjelasan:
File ini berisi seluruh endpoint (URL) yang digunakan aplikasi, baik untuk user maupun admin.

```
php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\DashboardController;
use App\Http\Controllers\Auth\LoginController;
use App\Services\FirebaseService;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Auth;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\SettingsController;

// Welcome page route
Route::get('/', [HomeController::class, 'index'])->name('welcome');

use App\Http\Controllers\FirebaseAuthController;

// Authentication routes
Route::middleware('guest')->group(function () {
    Route::get('login', [LoginController::class, 'showLoginForm'])->name('login');
    Route::post('login', [LoginController::class, 'login']);
    Route::post('firebase-login', [FirebaseAuthController::class, 'handle'])->name('firebase.login');
});

Route::post('logout', [LoginController::class, 'logout'])->name('logout');

// Protected routes (authenticated users)
Route::middleware('auth')->group(function () {
    // Dashboard
    Route::get('/dashboard', [DashboardController::class, 'index'])->name('dashboard');
    Route::post('/control', [DashboardController::class, 'control'])->name('control');

    // Views with role-based access
    Route::get('/history', function () {
        $user = auth()->user();
        if ($user && $user->role === 'admin') {
            return view('history');
        } else {
            return view('history-user');
        }
    })->name('history');

    Route::get('/controls', function () {
        $user = auth()->user();
        if ($user && $user->role === 'admin') {
return view('controls');
} else {
```

Kode Sumber

```
return view('controls-user');
}

})->name('controls');

Route::get('/schedules', [DashboardController::class, 'schedules'])->name('schedules');

// Settings & Profile
Route::get('/settings', [SettingsController::class, 'index'])->name('settings.index');
Route::put('/settings', [SettingsController::class, 'update'])->name('settings.update');
Route::get('/profile', [ProfileController::class, 'show'])->name('profile.show');
Route::get('/profile/edit', [ProfileController::class, 'edit'])->name('profile.edit');
Route::put('/profile', [ProfileController::class, 'update'])->name('profile.update');
Route::post('/profile/update-image', [ProfileController::class, 'updateImage'])->name('profile.update-image');
Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

// Admin routes with role middleware
Route::group(['prefix' => 'admin', 'middleware' => ['auth', \App\Http\Middleware\CheckRole::class . ':admin']], function () {
Route::get('/dashboard', [DashboardController::class, 'adminDashboard'])->name('admin.dashboard');

// User management routes
Route::resource('users', UserController::class)->names([
'index' => 'admin.users.index',
'create' => 'admin.users.create',
'store' => 'admin.users.store',
'edit' => 'admin.users.edit',
'update' => 'admin.users.update',
'destroy' => 'admin.users.destroy',
]);
});

// Sensor Data & Firebase Test
Route::get('/sensor-data', [DashboardController::class, 'sensorData'])->name('sensor.data');
Route::get('/firebase-test', function (FirebaseService $firebase) {
$data = $firebase->getSensorData();
return response()->json($data);
});

// Disable registration
Route::get('/register', function () {
abort(404);
});
```

Kode Sumber

app/Http/Controllers/

Penjelasan:
Controller utama yang menangani logika aplikasi, termasuk controller untuk dashboard, user management, dan API.

a. DashboardController.php

```
php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Services\FirebaseService;
use Illuminate\Support\Facades\Auth;

class DashboardController extends Controller
{
    protected $firebaseService;

    public function __construct(FirebaseService $firebaseService)
    {
        $this->firebaseService = $firebaseService;
        $this->middleware('auth');
    }

    public function index()
    {
        $user = auth()->user();

        if ($user->role === 'admin') {
            try {
                // Get sensor data from Firebase dashboard folder
                $sensorData = $this->firebaseService->getDatabase()
                    ->getReference('dashboard/sensor')
                    ->getValue();

                // Get command status from dashboard folder
                $commandData = $this->firebaseService->getDatabase()
                    ->getReference('dashboard/control')
                    ->getValue();

                // Get system status from dashboard folder
                $statusData = $this->firebaseService->getDatabase()
                    ->getReference('dashboard/status')
                    ->getValue();

                return view('dashboard', [
                    'sensorData' => $sensorData,
                    'commandData' => $commandData,
                    'statusData' => $statusData
                ]);
```

Kode Sumber

```
} catch (\Exception $e) {
\Log::error('Error fetching data from Firebase: ' . $e->getMessage());
return view('dashboard', [
'sensorData' => null,
'commandData' => null,
'statusData' => null
]);
}
} elseif ($user->role === 'user') {
try {
// Get sensor data from Firebase dashboard folder
$sensorData = $this->firebaseService->getDatabase()
->getReference('dashboard/sensor')
->getValue();

// Get system status from dashboard folder
$statusData = $this->firebaseService->getDatabase()
->getReference('dashboard/status')
->getValue();

return view('dashboard-user', [
'sensorData' => $sensorData,
'statusData' => $statusData
]);
} catch (\Exception $e) {
\Log::error('Error fetching data from Firebase: ' . $e->getMessage());
return view('dashboard-user', [
'sensorData' => null,
'statusData' => null
]);
}
} else {
abort(403, 'Unauthorized');
}
}

public function control(Request $request)
{
$user = auth()->user();
if (!$user || $user->role !== 'admin') {
return back()->with('error', 'Unauthorized access.');
}

$device = $request->input('device');
$status = $request->input('status');

if (!$device || !$status) {
return back()->with('error', 'Invalid device or status.');
}

// Map device names to Firebase paths
$deviceMap = [
'feeding' => 'aquarium/command/pakan',
'filter' => 'aquarium/command/pompa',
'lamp' => 'aquarium/command/lampu'
];

// Map status values to boolean
$statusMap = [
```

Kode Sumber

```
'on' => true,
'off' => false
];

if (!isset($deviceMap[$device]) || !isset($statusMap[$status])) {
return back()->with('error', 'Invalid device or status value.');
}

try {
$this->firebaseService->getDatabase()
->getReference($deviceMap[$device])
->set($statusMap[$status]);

// Simpan ke history/controls
$log = [
'action' => $device,
'status' => $status,
'user' => Auth::user() ? Auth::user()->email : 'unknown',
'timestamp' => round(microtime(true) * 1000)
];
$this->firebaseService->getDatabase()
->getReference('dashboard/history/controls/' . $log['timestamp'])
->set($log);

$statusMessage = [
'feeding' => 'Pakan berhasil diberikan',
'filter' => $status === 'on' ? 'Filter berhasil dinyalakan' : 'Filter berhasil dimatikan',
'lamp' => $status === 'on' ? 'Lampu berhasil dinyalakan' : 'Lampu berhasil dimatikan'
];

return back()->with('status', $statusMessage[$device]);
} catch (\Exception $e) {
\Log::error('Firebase command error: ' . $e->getMessage());
return back()->with('error', 'Gagal mengirim perintah ke perangkat.');
}
}

public function sensorData()
{
try {
$sensorData = $this->firebaseService->getDatabase()
->getReference('dashboard/sensor')
->getValue();
return view('sensor-data', compact('sensorData'));
} catch (\Exception $e) {
\Log::error('Error fetching sensor data from Firebase: ' . $e->getMessage());
return view('sensor-data', ['sensorData' => null]);
}
}

public function schedules()
{
$user = auth()->user();
if ($user && $user->role === 'admin') {
return view('schedules');
} else {
return view('schedules-user');
}
}
}
```

Kode Sumber

app/Http/Controllers/

b. UserController.php

```
php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rule;
use Kreait\Firebase\Factory;

class UserController extends Controller
{
    private $database;
    private $auth;

    public function __construct()
    {
        $factory = (new Factory)
            ->withServiceAccount(storage_path('app/firebase/firebase_credentials.json'))
            ->withDatabaseUri('https://smart-aquarium-3720d-default-rtdb.asia-southeast1.firebasedatabase.app/');
        $this->database = $factory->createDatabase();
        $this->auth = $factory->createAuth();
    }

    private function logToFirebase($action, $status, $details)
    {
        $now = now();
        $timestamp = $now->format('d-m-Y_H:i:s');
        $userEmail = auth()->user() ? auth()->user()->email : 'anonymous';

        $this->database->getReference('dashboard/history/Kontrol/' . $timestamp)
            ->set([
                'timestamp' => $now->getTimestamp() * 1000,
                'action' => $action,
                'status' => $status,
                'details' => $details,
                'user' => $userEmail
            ]);
    }

    public function index()
    {
        $users = User::paginate(10);
        return view('admin.users.index', compact('users'));
    }

    public function create()
    {
        return view('admin.users.create');
    }

    public function store(Request $request)
```

Kode Sumber

```
{
$validated = $request->validate([
'name' => 'required|string|max:255',
'email' => 'required|email|unique:users,email',
'password' => 'required|string|min:6|confirmed',
'role' => ['required', Rule::in(['admin', 'user'])],
]);

try {
if ($validated['role'] === 'admin') {
// Create user in Firebase Authentication
$firebaseUser = $this->auth->createUser([
'email' => $validated['email'],
'password' => $validated['password'],
'displayName' => $validated['name'],
]);

// Create user in Laravel database
$user = new User();
$user->name = $validated['name'];
$user->email = $validated['email'];
$user->password = Hash::make($validated['password']);
$user->role = $validated['role'];
$user->firebase_uid = $firebaseUser->uid;
$user->save();

$this->logToFirebase(
'Create User',
'Success',
"Created new admin user: {$user->name} ({$user->email}) with Firebase UID: {$user->firebase_uid}"
);
} else {
// Role is user, create only in Laravel database
$user = new User();
$user->name = $validated['name'];
$user->email = $validated['email'];
$user->password = Hash::make($validated['password']);
$user->role = $validated['role'];
$user->save();

$this->logToFirebase(
'Create User',
'Success',
"Created new user: {$user->name} ({$user->email})"
);
}

return redirect()->route('admin.users.index')
->with('success', 'User created successfully.');
} catch (\Exception $e) {
$this->logToFirebase(
'Create User',
'Failed',
"Failed to create user: {$validated['email']}. Error: {$e->getMessage()}"
);

return back()->with('error', 'Failed to create user: ' . $e->getMessage());
}
}
```

Kode Sumber

```
public function edit(User $user)
{
return view('admin.users.edit', compact('user'));
}

public function update(Request $request, User $user)
{
$validated = $request->validate([
'name' => 'required|string|max:255',
'email' => ['required', 'email', Rule::unique('users')->ignore($user->id)],
'password' => 'nullable|string|min:6|confirmed',
'role' => ['required', Rule::in(['admin', 'user'])],
]);

try {
$oldData = [
'name' => $user->name,
'email' => $user->email,
'role' => $user->role
];

$user->name = $validated['name'];
$user->email = $validated['email'];
if (!empty($validated['password'])) {
$user->password = Hash::make($validated['password']);
}
$user->role = $validated['role'];
$user->save();

$changes = [];
if ($oldData['name'] !== $user->name) $changes[] = "Name: {$oldData['name']} → {$user->name}";
if ($oldData['email'] !== $user->email) $changes[] = "Email: {$oldData['email']} → {$user->email}";
if ($oldData['role'] !== $user->role) $changes[] = "Role: {$oldData['role']} → {$user->role}";
if (!empty($validated['password'])) $changes[] = "Password: Updated";

$this->logToFirebase(
'Update User',
'Success',
"Updated user {$user->name} ({$user->email}). Changes: " . implode(', ', $changes)
);

return redirect()->route('admin.users.index')
->with('success', 'User updated successfully.');
} catch (\Exception $e) {
$this->logToFirebase(
'Update User',
'Failed',
"Failed to update user {$user->name} ({$user->email}). Error: {$e->getMessage()}"
);

return back()->with('error', 'Failed to update user: ' . $e->getMessage());
}
}

public function destroy(User $user)
{
try {
$userData = [
'name' => $user->name,
'email' => $user->email,
```

Kode Sumber

```
'role' => $user->role
];

$user->delete();

$this->logToFirebase(
'Delete User',
'Success',
"Deleted user: {$userData['name']} ({$userData['email']}) with role: {$userData['role']}"
);

return redirect()->route('admin.users.index')
->with('success', 'User deleted successfully.');
} catch (\Exception $e) {
$this->logToFirebase(
'Delete User',
'Failed',
"Failed to delete user {$user->name} ({$user->email}). Error: {$e->getMessage()}"
);

return back()->with('error', 'Failed to delete user: ' . $e->getMessage());
}
}
}
```

Kode Sumber

app/Http/Controllers/

c. Api/SensorController.php

```
php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Services\FirebaseService;
use Illuminate\Http\Request;

class SensorController extends Controller
{
    protected $firebaseService;

    public function __construct(FirebaseService $firebaseService)
    {
        $this->firebaseService = $firebaseService;
    }

    public function getSensorData()
    {
        try {
            $data = $this->firebaseService->getSensorData();
            return response()->json($data);
        } catch (\Exception $e) {
            \Log::error('Error fetching sensor data: ' . $e->getMessage());
            return response()->json(['error' => 'Failed to fetch sensor data'], 500);
        }
    }
}
```

Kode Sumber

app/Http/Controllers/

d. HomeController.php

```
php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Services\FirebaseService;

class HomeController extends Controller
{
    protected $firebaseService;

    public function __construct(FirebaseService $firebaseService)
    {
        $this->firebaseService = $firebaseService;
    }

    public function index()
    {
        try {
            // Get sensor data from Firebase dashboard folder
            $sensorData = $this->firebaseService->getDatabase()
                ->getReference('dashboard/sensor')
                ->getValue();

            // Get system status from dashboard folder
            $statusData = $this->firebaseService->getDatabase()
                ->getReference('dashboard/status')
                ->getValue();

            // Get historical data from Firebase
            $historyData = $this->firebaseService->getDatabase()
                ->getReference('dashboard/history')
                ->getValue();

            return view('welcome', [
                'sensorData' => $sensorData,
                'statusData' => $statusData,
                'historyData' => $historyData
            ]);
        } catch (\Exception $e) {
            \Log::error('Error fetching data from Firebase: ' . $e->getMessage());
            return view('welcome', [
                'sensorData' => null,
                'statusData' => null,
                'historyData' => null
            ]);
        }
    }

    public function dashboard()
    {
        $this->middleware('auth');
```

Kode Sumber

```
return view('home');
}
}
```

Kode Sumber

app/Http/Controllers/

e. SettingsController.php

```
php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Kreait\Firebase\Factory;

class SettingsController extends Controller
{
    public function index()
    {
        $user = Auth::user();
        return view('settings', [
            'user' => $user,
            'settings' => [
                'language' => $user->settings->language ?? 'en',
                'timezone' => $user->settings->timezone ?? 'Asia/Jakarta',
                'email_notifications' => $user->settings->email_notifications ?? true,
                'push_notifications' => $user->settings->push_notifications ?? true,
            ]
        ]);
    }

    public function update(Request $request)
    {
        $validated = $request->validate([
            'language' => 'required|in:en,id',
            'timezone' => 'required|in:Asia/Jakarta,Asia/Singapore',
            'email_notifications' => 'nullable|boolean',
            'push_notifications' => 'nullable|boolean',
        ]);

        $validated['email_notifications'] = $request->has('email_notifications');
        $validated['push_notifications'] = $request->has('push_notifications');

        $user = Auth::user();

        // Update user settings
        $user->settings()->updateOrCreate(
            ['user_id' => $user->id],
            $validated
        );

        // Catat ke Firebase history
        $factory = (new Factory)
            ->withServiceAccount(storage_path('app/firebase/firebase_credentials.json'))
            ->withDatabaseUri('https://smart-aquarium-3720d-default-rtdb.asia-southeast1.firebasedatabase.app/');
        $database = $factory->createDatabase();
        $now = now();
        $userEmail = auth()->user() ? auth()->user()->email : 'anonymous';
        $database->getReference('dashboard/history/notifikasi/' . $now->format('d-m-Y_H:i:s'))
```

Kode Sumber

```
->set([
'timestamp' => $now->getTimestamp() * 1000,
'type' => 'success',
'title' => 'Update Setting',
'message' => 'Pengaturan aplikasi berhasil diperbarui',
'user' => $userEmail
]);

return redirect()->route('settings.index')
->with('success', 'Settings updated successfully.');
}
}
```

Kode Sumber

app/Http/Controllers/

f. ProfileController.php

```
php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rule;
use Kreait\Firebase\Factory;

class ProfileController extends Controller
{
    public function show()
    {
        $user = Auth::user();
        return view('profile.show', compact('user'));
    }

    public function edit()
    {
        $user = Auth::user();
        return view('profile.edit', compact('user'));
    }

    public function update(Request $request)
    {
        $user = Auth::user();

        $validated = $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', Rule::unique('users')->ignore($user->id)],
            'current_password' => ['nullable', 'required_with:new_password', 'current_password'],
            'new_password' => ['nullable', 'min:8', 'confirmed'],
            'profile_image' => ['nullable', 'image', 'max:1024']
        ]);

        // Update basic info
        $user->name = $validated['name'];
        $user->email = $validated['email'];

        // Update password if provided
        if ($request->filled('new_password')) {
            $user->password = Hash::make($validated['new_password']);
        }

        // Handle profile image upload
        if ($request->hasFile('profile_image')) {
            // Delete old image if exists
            if ($user->profile_image && Storage::exists('public/profile/' . $user->profile_image)) {
                Storage::delete('public/profile/' . $user->profile_image);
            }
```

Kode Sumber

```
// Store new image
$imageName = time() . '.' . $request->profile_image->extension();
$request->profile_image->storeAs('public/profile', $imageName);
$user->profile_image = $imageName;
}

$user->save();

// Log to Firebase
$factory = (new Factory)
->withServiceAccount(storage_path('app/firebase/firebase_credentials.json'))
->withDatabaseUri('https://smart-aquarium-3720d-default-rtdb.asia-southeast1.firebasedatabase.app/');
$database = $factory->createDatabase();
$now = now();
$userEmail = auth()->user() ? auth()->user()->email : 'anonymous';
$database->getReference('dashboard/history/notifikasi/' . $now->format('d-m-Y_H:i:s'))
->set([
'timestamp' => $now->getTimestamp() * 1000,
'type' => 'success',
'title' => 'Update Profile',
'message' => 'Profile berhasil diperbarui',
'user' => $userEmail
]);

return redirect()->back()->with('success', 'Profile berhasil diperbarui');
}

public function updateImage(Request $request)
{
try {
$request->validate([
'profile_image' => ['required', 'image', 'max:1024']
]);

$user = Auth::user();

// Delete old image if exists
if ($user->profile_image && Storage::exists('public/profile/' . $user->profile_image)) {
Storage::delete('public/profile/' . $user->profile_image);
}

// Store new image
$imageName = time() . '_' . uniqid() . '.' . $request->profile_image->extension();
$path = $request->profile_image->storeAs('public/profile', $imageName);

if (!$path) {
throw new \Exception('Failed to store image');
}

$user->profile_image = $imageName;
$user->save();

return response()->json([
'success' => true,
'image_url' => Storage::url('profile/' . $imageName)
]);
} catch (\Exception $e) {
\Log::error('Profile image upload failed: ' . $e->getMessage());
return response()->json([
```

Kode Sumber

```
'success' => false,
'message' => 'Failed to upload image: ' . $e->getMessage()
], 500);
}
}
}
```

Kode Sumber

app/Http/Controllers/

g. FirebaseAuthController.php

```
php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use App\Services\FirebaseService;
use Illuminate\Support\Str;
use Illuminate\Support\Facades\Log;

class FirebaseAuthController extends Controller
{
    protected $firebaseService;

    public function __construct(FirebaseService $firebaseService)
    {
        $this->firebaseService = $firebaseService;
    }

    public function handle(Request $request)
    {
        $request->validate([
            'firebase_token' => 'required|string',
        ]);

        try {
            $verified = $this->firebaseService->verifyIdToken($request->firebase_token);
            if (!$verified) {
                return response()->json(['error' => 'Invalid Firebase token'], 401);
            }

            $uid = $verified->claims()->get('sub');
            $fbUser = $this->firebaseService->getAuth()->getUser($uid);

            $user = User::firstOrNew(['email' => $fbUser->email]);
            $user->name = $fbUser->displayName ?? $fbUser->email;
            $user->role = 'admin';
            $user->password = bcrypt(Str::random(16));
            $user->save();

            Auth::login($user);

            return response()->json(['status' => 'ok']);
        } catch (\Exception $e) {
            Log::error('Firebase login error: ' . $e->getMessage());
            return response()->json(['error' => 'Firebase login failed'], 500);
        }
    }
}
```

Kode Sumber

app/Http/Controllers/

h. Auth/LoginController.php

```
php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends Controller
{
    use AuthenticatesUsers;

    protected $redirectTo = RouteServiceProvider::HOME;

    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

Kode Sumber

app/Http/Controllers/

i. Auth/RegisterController.php

```
php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Foundation\Auth\RegistersUsers;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;

class RegisterController extends Controller
{
    use RegistersUsers;

    protected $redirectTo = '/dashboard';

    public function __construct()
    {
        $this->middleware('guest');
    }

    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
        ]);
    }

    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
            'role' => 'user',
        ]);
    }
}
```

Kode Sumber

app/Http/Controllers/

j. Auth/ForgotPasswordController.php

```
php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\SendsPasswordResetEmails;

class ForgotPasswordController extends Controller
{
    use SendsPasswordResetEmails;

    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

k. Auth/ResetPasswordController.php

```
php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ResetsPasswords;

class ResetPasswordController extends Controller
{
    use ResetsPasswords;

    protected $redirectTo = '/dashboard';

    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

Kode Sumber

app/Http/Controllers/

l. Auth/ConfirmPasswordController.php

```
php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ConfirmsPasswords;

class ConfirmPasswordController extends Controller
{
    use ConfirmsPasswords;

    protected $redirectTo = '/dashboard';

    public function __construct()
    {
        $this->middleware('auth');
    }
}
```

m. Auth/VerificationController.php

```
php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\VerifiesEmails;

class VerificationController extends Controller
{
    use VerifiesEmails;

    protected $redirectTo = '/dashboard';

    public function __construct()
    {
        $this->middleware('auth');
        $this->middleware('signed')->only('verify');
        $this->middleware('throttle:6,1')->only('verify', 'resend');
    }
}
```

Kode Sumber

app/Http/Middleware/

Penjelasan:
Middleware untuk autentikasi dan validasi khusus.

a. CheckRole.php

```
php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class CheckRole
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @param  string  $role
     * @return mixed
     */
    public function handle(Request $request, Closure $next, $role)
    {
        if (!Auth::check()) {
            return redirect()->route('login');
        }

        if (Auth::user()->role !== $role) {
            return redirect()->route('dashboard')->with('error', 'Unauthorized access.');
        }

        return $next($request);
    }
}
```

Kode Sumber

app/Http/Middleware/

b. FirebaseAuthMiddleware.php

```
php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class FirebaseAuthMiddleware
{
    /**
     * Handle an incoming request.
     * Allow only users authenticated via Firebase (e.g., role 'vip_user').
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle(Request $request, Closure $next)
    {
        $user = Auth::user();

        if (!$user || $user->role !== 'vip_user') {
            return redirect()->route('login')->with('error', 'Access denied. Please login with Firebase account.');
        }

        return $next($request);
    }
}
```

Kode Sumber

app/Models/

Penjelasan:
Model utama yang merepresentasikan tabel database.

a. User.php

```
php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
        'role',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var list<string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
```

Kode Sumber

app/Models/

```
}

public function settings()
{
return $this->hasOne(UserSettings::class);
}
}
```

Kode Sumber

app/Models/

b. UserSettings.php

```
php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class UserSettings extends Model
{
    protected $fillable = [
        'user_id',
        'language',
        'timezone',
        'email_notifications',
        'push_notifications',
    ];

    protected $casts = [
        'email_notifications' => 'boolean',
        'push_notifications' => 'boolean',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

Kode Sumber

app/Services/

Penjelasan:
Service layer untuk integrasi dengan Firebase dan logika bisnis.

a. FirebaseService.php

```
php
<?php

namespace App\Services;

use Kreait\Firebase\Factory;
use Kreait\Firebase\Database;
use Kreait\Firebase\Auth as FirebaseAuth;
use Kreait\Firebase\Exception\Auth\FailedToVerifyToken;
use Kreait\Firebase\Exception\DatabaseException;
use Kreait\Firebase\ServiceAccount;

class FirebaseService
{
    protected $firebase;
    protected $auth;

    public function __construct()
    {
        try {
            $credentialsPath = storage_path('app/firebase/firebase_credentials.json');
            if (!file_exists($credentialsPath)) {
                throw new \Exception('Firebase credentials file not found at: ' . $credentialsPath);
            }

            $credentials = json_decode(file_get_contents($credentialsPath), true);
            if (json_last_error() !== JSON_ERROR_NONE) {
                throw new \Exception('Invalid JSON in Firebase credentials file');
            }

            $factory = (new Factory)
                ->withServiceAccount($credentialsPath)
                ->withDatabaseUri('https://smart-aquarium-3720d-default-rtdb.asia-southeast1.firebasedatabase.app');

            $this->firebase = $factory->createDatabase();
            $this->auth = $factory->createAuth();

            \Log::info('Firebase connection established successfully');
        } catch (\Exception $e) {
            \Log::error('Firebase connection error: ' . $e->getMessage());
            throw $e;
        }
    }

    public function getDatabase()
    {
```

Kode Sumber

app/Services/

```
return $this->firebase;
}

public function getAuth()
{
return $this->auth;
}

public function verifyIdToken(string $idToken)
{
try {
$verifiedIdToken = $this->auth->verifyIdToken($idToken);
return $verifiedIdToken;
} catch (FailedToVerifyToken $e) {
\Log::error('Failed to verify Firebase ID token: ' . $e->getMessage());
return null;
}
}

public function getSensorData()
{
try {
$data = $this->firebase->getReference('aquarium/data')->getValue();
\Log::info('Retrieved sensor data', ['data' => $data]);

if (empty($data)) {
\Log::warning('No sensor data found, initializing...');
$this->initializeData();
$data = $this->firebase->getReference('aquarium/data')->getValue();
}

// Ensure all required keys exist
$defaultData = [
'suhu_air' => 0,
'suhu_udara' => 0,
'kelembapan' => 0,
'level_air' => 'Normal',
'waktu_update' => date('Y-m-d H:i:s')
];

$data = array_merge($defaultData, $data ?? []);

// Update level_air status based on value
if (isset($data['level_air']) && is_numeric($data['level_air'])) {
$settings = $this->firebase->getReference('aquarium/settings')->getValue();
$minLevel = $settings['batas_level_air_min'] ?? 20;
$maxLevel = $settings['batas_level_air_max'] ?? 80;

if ($data['level_air'] < $minLevel) {
$data['level_air'] = 'Rendah';
} elseif ($data['level_air'] > $maxLevel) {
$data['level_air'] = 'Tinggi';
} else {
$data['level_air'] = 'Normal';
}
```

Kode Sumber

app/Services/

```
}

return $data;
} catch (DatabaseException $e) {
\Log::error('Error fetching sensor data: ' . $e->getMessage());
return [
'suhu_air' => 0,
'suhu_udara' => 0,
'kelembapan' => 0,
'level_air' => 'Normal',
'waktu_update' => date('Y-m-d H:i:s')
];
}
}

public function updateCommand($path, $value)
{
try {
return $this->firebase->getReference($path)->set($value);
} catch (DatabaseException $e) {
\Log::error('Error updating command: ' . $e->getMessage());
return false;
}
}
}
```

Kode Sumber

app/Console/Commands/

Penjelasan:
Command untuk generate data sensor testing.

a. GenerateSensorData.php

```
php
<?php

namespace App\Console\Commands;

use Illuminate\Console\Command;

class GenerateSensorData extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'app:generate-sensor-data';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Command description';

    /**
     * Execute the console command.
     */
    public function handle()
    {
        $factory = (new \Kreait\Firebase\Factory())
            ->withServiceAccount(storage_path('app/firebase/firebase_credentials.json'))
            ->withDatabaseUri('https://smart-aquarium-3720d-default-rtdb.asia-southeast1.firebasedatabase.app');
        $db = $factory->createDatabase();

        for ($i = 0; $i < 100; $i++) {
            $data = [
                'suhu_ds18b20' => rand(250, 300) / 10,
                'suhu_bme280' => rand(260, 320) / 10,
                'kelembapan' => rand(50, 80),
                'tekanan' => rand(1000, 1020),
                'level_air' => rand(60, 90),
            ];
            $timestamp = now()->timestamp . $i;
            $db->getReference("dashboard/history/{$timestamp}")->set($data);
            $this->info("Data sent: " . json_encode($data));
            sleep(5); // interval 5 detik
        }
    }
}
```

Kode Sumber

database/migrations/

Penjelasan:
Struktur tabel database (cuplikan migration penting).

a. create_user_settings_table.php

```
php
<?php

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

return new class extends Migration
{
    public function up()
    {
        Schema::create('user_settings', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->onDelete('cascade');
            $table->string('language')->default('en');
            $table->string('timezone')->default('Asia/Jakarta');
            $table->boolean('email_notifications')->default(true);
            $table->boolean('push_notifications')->default(true);
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('user_settings');
    }
};
```

Kode Sumber

database/migrations/

b. add_role_to_users_table.php

```
php
<?php

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

return new class extends Migration
{
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->string('role')->default('user')->after('password');
        });
    }

    public function down()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn('role');
        });
    }
};
```

Kode Sumber

config/

Penjelasan:
File konfigurasi aplikasi.

a. firebase.php

```
php
<?php

return [
    'database_url' => env('FIREBASE_DATABASE_URL', 'https://smart-aquarium-3720d-default-rtdb.asia-southeast1.firebasedatabase.app'),
    'credentials_path' => env('FIREBASE_CREDENTIALS', storage_path('app/firebase/firebase_credentials.json')),
    'project_id' => env('FIREBASE_PROJECT_ID', 'smart-aquarium-3720d'),
    'storage_bucket' => env('FIREBASE_STORAGE_BUCKET', 'smart-aquarium-3720d.appspot.com'),
    'api_key' => env('FIREBASE_API_KEY'),
    'auth_domain' => env('FIREBASE_AUTH_DOMAIN', 'smart-aquarium-3720d.firebaseapp.com'),
    'messaging_sender_id' => env('FIREBASE_MESSAGING_SENDER_ID'),
    'app_id' => env('FIREBASE_APP_ID'),
];
```

Kode Sumber

composer.json & package.json

Penjelasan:
Menunjukkan dependency utama PHP dan JavaScript.

a. composer.json

```
json
{
    "name": "laravel/laravel",
    "type": "project",
    "description": "The skeleton application for the Laravel framework.",
    "keywords": ["laravel", "framework"],
    "license": "MIT",
    "require": {
        "php": "^8.2",
        "kreait/firebase-php": "^7.18",
        "laravel/framework": "^11.0",
        "laravel/sanctum": "^4.1",
        "laravel/tinker": "^2.9",
        "laravel/ui": "^4.6"
    },
    "require-dev": {
        "fakerphp/faker": "^1.23",
        "laravel/pint": "^1.13",
        "laravel/sail": "^1.26",
        "mockery/mockery": "^1.6",
        "nunomaduro/collision": "^8.0",
        "phpunit/phpunit": "^10.5",
        "spatie/laravel-ignition": "^2.4"
    }
}
```

b. package.json

```
json
{
    "private": true,
    "type": "module",
    "scripts": {
        "dev": "vite",
        "build": "vite build"
    },
    "devDependencies": {
        "@popperjs/core": "^2.11.6",
        "@vitejs/plugin-vue": "^4.5.0",
        "axios": "^1.6.4",
        "bootstrap": "^5.2.3",
        "laravel-vite-plugin": "^1.0",
        "sass": "^1.56.1",
        "vite": "^5.0",
        "vue": "^3.2.37"
    }
}
```

Kode Sumber

File khusus integrasi IoT/Firebase

Penjelasan:
File yang menghubungkan aplikasi dengan perangkat IoT/Firebase.

a. esp8266.ino

```
#include <ESP8266WiFi.h>
#include <Firebase_ESP_Client.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <Adafruit_BME280.h>
#include <Servo.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <time.h>

// — WiFi & Firebase Credentials —
#define WIFI_SSID        "sul"
#define WIFI_PASSWORD    "123456789"
#define API_KEY          "AIzaSyC6zxY_ljbhoQEMbZYHuDRNZ2GGUbswQes"
#define DATABASE_URL     "https://smart-aquarium-3720d-default-rtdb.asia-
southeast1.firebasedatabase.app/"
#define FIREBASE_EMAIL   "project@gmail.com"
#define FIREBASE_PASSWORD "123456"

// — Firebase Paths —
#define PATH_SENSOR      "/dashboard/sensor"
#define PATH_HISTORY     "/dashboard/history/sensor"
#define PATH_FEED_CONTROL "/dashboard/control/pakan"
#define PATH_LAST_FEED   "/dashboard/status/last_feed"
#define PATH_BUZZER_STAT  "/dashboard/status/buzzer"
#define PATH_WL_LOW      "/dashboard/status/water_level_low"
#define PATH_WL_HIGH     "/dashboard/status/water_level_high"

// — Pins —
#define DS18B20_PIN      14  // D5
#define SDA_PIN          5  // D1
#define SCL_PIN          4  // D2
#define SERVO_PIN        16  // D0
#define TRIG_PIN         13  // D7
#define ECHO_PIN         12  // D6
#define BUZZER_PIN       2  // D4
#define BUTTON_PIN       15  // D8

#define WATER_LEVEL_MAX   3
#define WATER_LEVEL_MIN   10

// — Firebase & Networking —
FirebaseData fbdo;
FirebaseAuth auth;
```

Kode Sumber

File khusus integrasi IoT/Firebase

```
FirebaseConfig config;
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 7*3600, 60000);

// — Sensors & Actuators —
OneWire oneWire(DS18B20_PIN);
DallasTemperature ds18b20(&oneWire);
Adafruit_BME280 bme;
Servo feedingServo;

// — State —
unsigned long lastSensorUpdate = 0;
const unsigned long SENSOR_INTERVAL = 60000;  // 1 menit
bool buzzerEnabled = true;
int  waterLevelLow  = 20;
int  waterLevelHigh = 80;

// — Feeding Control —
unsigned long lastFeedTime      = 0;
const unsigned long feedCooldown = 1000;    // 10 detik
static bool lastButtonState     = HIGH;
static bool lastFirebaseState   = false;

void setup() {
 Serial.begin(115200);
 delay(200);

 // pin modes
 pinMode(TRIG_PIN, OUTPUT);
 pinMode(ECHO_PIN, INPUT);
 pinMode(BUZZER_PIN, OUTPUT);
 pinMode(BUTTON_PIN, INPUT_PULLUP);

 // servo init
 feedingServo.attach(SERVO_PIN);
 delay(200);
 feedingServo.write(90);

 // WiFi
 WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
 Serial.print("WiFi ");
 while (WiFi.status() != WL_CONNECTED) {
  delay(300); Serial.print(".");
 }
 Serial.println(" connected");

 // configure local time for history timestamps
 configTime(7*3600, 0, "pool.ntp.org");

 // Firebase & NTPClient
 timeClient.begin();
 config.api_key      = API_KEY;
 config.database_url = DATABASE_URL;
 auth.user.email    = FIREBASE_EMAIL;
 auth.user.password = FIREBASE_PASSWORD;
```

Kode Sumber

File khusus integrasi IoT/Firebase

```
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);

 // sensors
 ds18b20.begin();
 Wire.begin(SDA_PIN, SCL_PIN);
 Wire.setClock(100000);
 if (!bme.begin(0x76) && !bme.begin(0x77)) {
  Serial.println("BME280 init failed");
  while (1) delay(500);
 }

 // reset feed flag
 Firebase.RTDB.setBool(&fbdo, PATH_FEED_CONTROL, false);

 Serial.println("Setup done");
}

void loop() {
 unsigned long now = millis();

 // 1) Periodic sensor update + history
 if (now - lastSensorUpdate >= SENSOR_INTERVAL) {
  lastSensorUpdate = now;
  float t1 = readDS18B20();
  float t2 = bme.readTemperature();
  float h  = bme.readHumidity();
  float p  = bme.readPressure() / 100.0F;
  int   wl = readWaterLevel();

  updateFirebaseData(t1,t2,h,p,wl);
  saveSensorHistory(t1,t2,h,p,wl);
  if (buzzerEnabled) checkWaterLevel(wl);
 }

 // 2) Physical button — edge detect
 bool btn = digitalRead(BUTTON_PIN);
 if (btn == LOW && lastButtonState == HIGH && now - lastFeedTime > feedCooldown) {
  Serial.println("Button feed");
  activateFeeding();
  lastFeedTime = now;
 }
 lastButtonState = btn;

 // 3) Firebase control — edge detect
 if (Firebase.RTDB.getBool(&fbdo, PATH_FEED_CONTROL)) {
  bool fbState = fbdo.to<bool>();
  if (fbState && !lastFirebaseState && now - lastFeedTime > feedCooldown) {
   Serial.println("Firebase feed");
   activateFeeding();
   lastFeedTime = now;
  }
  lastFirebaseState = fbState;
 }
}
```

Kode Sumber

File khusus integrasi IoT/Firebase

```
float readDS18B20() {
  ds18b20.requestTemperatures();
  return ds18b20.getTempCByIndex(0);
}

int readWaterLevel() {
  digitalWrite(TRIG_PIN, LOW);  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long dur = pulseIn(ECHO_PIN, HIGH);
  float dist = dur * 0.034 / 2.0;
  return constrain(map(dist,WATER_LEVEL_MIN, WATER_LEVEL_MAX,100,0),0,100);
}

void checkWaterLevel(int lvl) {
  if (Firebase.RTDB.getInt(&fbdo, PATH_WL_LOW))
    waterLevelLow  = fbdo.to<int>();
  if (Firebase.RTDB.getInt(&fbdo, PATH_WL_HIGH))
    waterLevelHigh = fbdo.to<int>();
  if (lvl < waterLevelLow)      tone(BUZZER_PIN,1000,500);
  else if (lvl > waterLevelHigh) tone(BUZZER_PIN,2000,500);
}

void updateFirebaseData(float t1,float t2,float h,float p,int wl) {
  Firebase.RTDB.setFloat(&fbdo, PATH_SENSOR "/suhu_ds18b20", t1);
  Firebase.RTDB.setFloat(&fbdo, PATH_SENSOR "/suhu_bme280",  t2);
  Firebase.RTDB.setFloat(&fbdo, PATH_SENSOR "/kelembapan",   h);
  Firebase.RTDB.setFloat(&fbdo, PATH_SENSOR "/tekanan",      p);
  Firebase.RTDB.setInt(  &fbdo, PATH_SENSOR "/level_air",    wl);
  Firebase.RTDB.setBool( &fbdo, PATH_BUZZER_STAT, buzzerEnabled);
}

void saveSensorHistory(float t1, float t2, float h, float p, int wl) {
  // Ambil waktu lokal
  time_t now = time(nullptr);
  struct tm *ptm = localtime(&now);

  // Format: dd-mm-yyyy_hh:mm:ss
  char buf[25];
  sprintf(buf, "%02d-%02d-%d_%02d:%02d:%02d",
          ptm->tm_mday, ptm->tm_mon + 1, ptm->tm_year + 1900,
          ptm->tm_hour, ptm->tm_min, ptm->tm_sec);

  String path = String(PATH_HISTORY) + "/" + buf;

  Firebase.RTDB.setFloat(&fbdo, path + "/suhu_ds18b20", t1);
  Firebase.RTDB.setFloat(&fbdo, path + "/suhu_bme280",  t2);
  Firebase.RTDB.setFloat(&fbdo, path + "/kelembapan",   h);
  Firebase.RTDB.setFloat(&fbdo, path + "/tekanan",      p);
  Firebase.RTDB.setInt(  &fbdo, path + "/level_air",    wl);
}
```

Kode Sumber

File khusus integrasi IoT/Firebase

```
void activateFeeding() {
 feedingServo.write(115); delay(700);
 feedingServo.write(90);  delay(200);
 Firebase.RTDB.setBool(&fbdo, PATH_FEED_CONTROL, false);

 // Log waktu pakan terakhir
 time_t now = time(nullptr);
 struct tm *ptm = localtime(&now);
 char buf[20];
 sprintf(buf, "%02d:%02d:%02d", ptm->tm_hour, ptm->tm_min, ptm->tm_sec);
 Firebase.RTDB.setString(&fbdo, PATH_LAST_FEED, buf);
 Serial.println("Fed at " + String(buf));
}
```

Kode Sumber

.env

Penjelasan:
Berkas konfigurasi environment digunakan untuk menyimpan pengaturan yang bersifat sensitif atau khusus untuk setiap lingkungan.

```
APP_NAME="Smart Aquarium"
APP_ENV=local
APP_KEY=base64:PGwfO3bYJvWCzySxyOoS/L/QobELTCyNE10Kc7Irw4I=
APP_DEBUG=true
APP_URL=http://localhost

APP_LOCALE=en
APP_FALLBACK_LOCALE=en
APP_FAKER_LOCALE=en_US

APP_MAINTENANCE_DRIVER=file
# APP_MAINTENANCE_STORE=database

PHP_CLI_SERVER_WORKERS=4

BCRYPT_ROUNDS=12

LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=smart-aquarium
DB_USERNAME=root
DB_PASSWORD=

FIREBASE_CREDENTIALS=/storage/app/firebase/firebase_credentials.json
FIREBASE_DATABASE_URL=https://smart-aquarium-3720d-default-rtdb.asia-
southeast1.firebasedatabase.app

SESSION_DRIVER=database
SESSION_LIFETIME=120
SESSION_ENCRYPT=false
SESSION_PATH=/
SESSION_DOMAIN=null

BROADCAST_CONNECTION=log
FILESYSTEM_DISK=local
QUEUE_CONNECTION=database

CACHE_STORE=database
# CACHE_PREFIX=
```

Kode Sumber

.env

```
MEMCACHED_HOST=127.0.0.1

REDIS_CLIENT=phpredis
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=log
MAIL_SCHEME=null
MAIL_HOST=127.0.0.1
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

VITE_APP_NAME="${APP_NAME}"
```

Kode Sumber

resources/views/

Penjelasan:
Tampilan utama aplikasi (Blade template) yang mewakili antarmuka pengguna dan fitur utama.

a. layouts/app.blade.php (Layout Utama)

```
blade
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>{{ config('app.name', 'Smart Aquarium') }}</title>
    <link rel="stylesheet" href="{{ asset('css/app.css') }}">
    <script src="{{ asset('js/app.js') }}" defer></script>
    <!-- ... Bootstrap, FontAwesome, Custom CSS ... -->
</head>
<body>
    <div class="aquarium-bg"> ... </div>
    <div id="app">
        @yield('content')
    </div>
</body>
</html>
```

b. welcome.blade.php (Landing Page)

```
blade
@extends('layouts.app')
@section('content')
<div class="welcome-container">
    <h1>Selamat Datang di Smart Aquarium</h1>
    <p>Smart Aquarium adalah sistem pemantauan dan kontrol akuarium berbasis web dan IoT.</p>
    <!-- Tampilkan data sensor dan status secara real-time -->
    <div class="sensor-summary"> ... </div>
</div>
@endsection
```

c. home.blade.php (Dashboard User)

```
blade
@extends('layouts.app')
@section('content')
<div class="dashboard-user">
    <h2>Dashboard User</h2>
    <!-- Ringkasan data sensor, status, dan notifikasi -->
    <div class="sensor-cards"> ... </div>
</div>
@endsection
```

Kode Sumber

resources/views/

d. settings.blade.php (Pengaturan)

```
blade
@extends('layouts.app')
@section('content')
<div class="settings-page">
  <h2>Pengaturan Akun</h2>
  <form method="POST" action="{{ route('settings.update') }}">
    @csrf
    @method('PUT')
    <!-- Form pengaturan bahasa, zona waktu, notifikasi -->
    <button type="submit">Simpan</button>
  </form>
</div>
@endsection
```

e. controls.blade.php (Kontrol Perangkat)

```
blade
@extends('layouts.app')
@section('content')
<div class="controls-page">
  <h2>Kontrol Perangkat Akuarium</h2>
  <!-- Tombol kontrol pakan, filter, lampu, dsb -->
  <form method="POST" action="{{ route('control') }}">
    @csrf
    <!-- Pilihan perangkat dan status -->
    <button type="submit">Kirim Perintah</button>
  </form>
</div>
@endsection
```

f. history.blade.php (Riwayat)

```
blade
@extends('layouts.app')
@section('content')
<div class="history-page">
  <h2>Riwayat Data Sensor & Aktivitas</h2>
  <!-- Tabel/grafik riwayat data sensor dan log aktivitas -->
  <table class="table"> ... </table>
</div>
@endsection
```

Kode Sumber

resources/views/

g. schedules.blade.php (Jadwal)

```
blade
@extends('layouts.app')
@section('content')
<div class="schedules-page">
  <h2>Jadwal Otomatis</h2>
  <!-- Form tambah/edit jadwal, daftar jadwal aktif -->
  <form> ... </form>
  <table class="table"> ... </table>
</div>
@endsection
```

h. sensor-data.blade.php (Data Sensor)

```
blade
@extends('layouts.app')
@section('content')
<div class="sensor-data-page">
  <h2>Data Sensor Real-time</h2>
  <!-- Tabel/grafik data sensor terkini -->
  <div class="sensor-table"> ... </div>
</div>
@endsection
```

i. dashboard-user.blade.php (Dashboard User)

```
blade
@extends('layouts.app')
@section('content')
<div class="dashboard-user">
  <h2>Dashboard User</h2>
  <!-- Ringkasan data sensor, status, dan kontrol user -->
  <div class="sensor-cards"> ... </div>
</div>
@endsection
```

j. auth/login.blade.php (Login)

```
blade
@extends('layouts.app')
@section('content')
<div class="login-container">
  <h2>Login</h2>
  <form method="POST" action="{{ route('login') }}">
    @csrf
    <input type="email" name="email" required>
    <input type="password" name="password" required>
    <button type="submit">Login</button>
  </form>
</div>
@endsection
```

Kode Sumber

resources/views/

k. auth/register.blade.php (Registrasi)

```
blade
@extends('layouts.app')
@section('content')
<div class="register-container">
  <h2>Registrasi</h2>
  <form method="POST" action="{{ route('register') }}">
    @csrf
    <input type="text" name="name" required>
    <input type="email" name="email" required>
    <input type="password" name="password" required>
    <input type="password" name="password_confirmation" required>
    <button type="submit">Daftar</button>
  </form>
</div>
@endsection
```

l. auth/passwords/reset.blade.php (Reset Password)

```
blade
@extends('layouts.app')
@section('content')
<div class="reset-password-container">
  <h2>Reset Password</h2>
  <form method="POST" action="{{ route('password.update') }}">
    @csrf
    <input type="hidden" name="token" value="{{ $token }}">
    <input type="email" name="email" required>
    <input type="password" name="password" required>
    <input type="password" name="password_confirmation" required>
    <button type="submit">Reset Password</button>
  </form>
</div>
@endsection
```

Kode Sumber

resources/views/

m. admin/users/index.blade.php (Daftar User Admin)

```
blade
@extends('layouts.app')
@section('content')
<div class="admin-users-index">
  <h2>Daftar User</h2>
  <a href="{{ route('admin.users.create') }}">Tambah User</a>
  <table class="table">
    <thead><tr><th>Nama</th><th>Email</th><th>Role</th><th>Aksi</th></tr></thead>
    <tbody>
      @foreach($users as $user)
      <tr>
        <td>{{ $user->name }}</td>
        <td>{{ $user->email }}</td>
        <td>{{ $user->role }}</td>
        <td>
          <a href="{{ route('admin.users.edit', $user) }}">Edit</a>
          <form method="POST" action="{{ route('admin.users.destroy', $user) }}" style="display:inline">
            @csrf @method('DELETE')
            <button type="submit">Hapus</button>
          </form>
        </td>
      </tr>
      @endforeach
    </tbody>
  </table>
</div>
@endsection
```

n. admin/users/create.blade.php (Tambah User Admin)

```
blade
@extends('layouts.app')
@section('content')
<div class="admin-users-create">
  <h2>Tambah User</h2>
  <form method="POST" action="{{ route('admin.users.store') }}">
    @csrf
    <input type="text" name="name" required>
    <input type="email" name="email" required>
    <input type="password" name="password" required>
    <input type="password" name="password_confirmation" required>
    <select name="role">
      <option value="user">User</option>
      <option value="admin">Admin</option>
    </select>
    <button type="submit">Simpan</button>
  </form>
</div>
@endsection
```

Kode Sumber

resources/views/

o. admin/users/edit.blade.php (Edit User Admin)

```
blade
@extends('layouts.app')
@section('content')
<div class="admin-users-edit">
  <h2>Edit User</h2>
  <form method="POST" action="{{ route('admin.users.update', $user) }}">
    @csrf @method('PUT')
    <input type="text" name="name" value="{{ $user->name }}" required>
    <input type="email" name="email" value="{{ $user->email }}" required>
    <select name="role">
      <option value="user" @if($user->role=='user') selected @endif>User</option>
      <option value="admin" @if($user->role=='admin') selected @endif>Admin</option>
    </select>
    <button type="submit">Update</button>
  </form>
</div>
@endsection
```

p. profile/show.blade.php (Profil)

```
blade
@extends('layouts.app')
@section('content')
<div class="profile-show">
  <h2>Profil Pengguna</h2>
  <p>Nama: {{ $user->name }}</p>
  <p>Email: {{ $user->email }}</p>
  <a href="{{ route('profile.edit') }}">Edit Profil</a>
</div>
@endsection
```

q. profile/edit.blade.php (Edit Profil)

```
blade
@extends('layouts.app')
@section('content')
<div class="profile-edit">
  <h2>Edit Profil</h2>
  <form method="POST" action="{{ route('profile.update') }}" enctype="multipart/form-data">
    @csrf @method('PUT')
    <input type="text" name="name" value="{{ $user->name }}" required>
    <input type="email" name="email" value="{{ $user->email }}" required>
    <input type="password" name="new_password">
    <input type="password" name="new_password_confirmation">
    <input type="file" name="profile_image">
    <button type="submit">Simpan</button>
  </form>
</div>
@endsection
```