Islam360

Religious App — API & Infrastructure

Complete Setup Documentation

AWS S3 · CloudFront · Aurora MySQL · Laravel API

Version 1.0 | 2026

1. Project Overview

Islam360 is a religious content application that serves Islamic content (Quran, Hadith, Duas, Tasbihaat) to mobile and web users via a Laravel REST API. The infrastructure is built on AWS using EC2, S3, CloudFront, and Aurora MySQL Serverless.

1.1 Architecture Overview

The complete request flow for the application is:

```
Mobile/Web App
    │
    ├──▶ Laravel API (EC2 + Docker)
    │         │
    │         └──▶ Aurora MySQL Serverless (text data + CloudFront URLs)
    │
    └──▶ CloudFront CDN (audio, images, PDFs)
               │
               └──▶ S3 Bucket (private — CloudFront only)
```

1.2 Why This Separation?

Data Type	Where & Why
Text Data (Quran, Hadith, Duas)	Aurora MySQL — structured, searchable, fast queries
Audio Files (mp3, recitations)	S3 — cheap storage built for large files
Images, PDFs	S3 — same reason as audio
Content Delivery	CloudFront CDN — serves globally from nearest edge location
API / Business Logic	EC2 Laravel — handles requests, authentication, logic

2. AWS S3 Bucket Setup

S3 stores all media files (audio, images, PDFs). The bucket is kept private — only CloudFront can access it.

Step 1 — Create S3 Bucket

1. Go to AWS Console → S3 → Click Create Bucket
2. Bucket name: islam360-content (or your preferred name)
3. Region: ap-south-1 (Mumbai) — closest to Pakistan, UAE, Middle East users
4. Block all public access: ON — keep bucket private
5. Click Create Bucket

⚠ **Note:** Keep the bucket completely private. Users will never access S3 directly. All file delivery goes through CloudFront.

Step 2 — Create IAM User for Laravel

6. Go to AWS Console → IAM → Users → Create User
7. Username: islam360-laravel
8. Attach policy: AmazonS3FullAccess
9. Create user → Go to Security Credentials tab
10. Create Access Key → Select: Application running on EC2
11. Save Access Key ID and Secret Access Key — you will add these to .env

⚠ **Note:** Never commit your AWS keys to Git. Always store them in .env file which is in .gitignore.

3. AWS CloudFront Setup

CloudFront is a CDN that sits in front of S3. It caches your files globally so users in Pakistan, UAE, UK and everywhere get fast delivery from the nearest edge location.

Step 1 — Create CloudFront Distribution

12. Go to AWS Console → CloudFront → Create Distribution
13. Origin Domain: select your S3 bucket (islam360-content.s3.amazonaws.com)
14. Origin Access: select Origin Access Control (OAC) — keeps S3 private
15. Create OAC → give it a name → Create
16. Default Cache Behavior → Viewer Protocol Policy: Redirect HTTP to HTTPS
17. Click Create Distribution
18. Wait 5-10 minutes for deployment to complete
19. Copy your CloudFront domain — example: d1234abcd.cloudfront.net

Step 2 — Update S3 Bucket Policy for CloudFront

After creating CloudFront it will show you a bucket policy. Go to S3 → your bucket → Permissions → Bucket Policy and paste:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::islam360-content/*"
    }
  ]
}
```

ℹ This policy means ONLY CloudFront can read from S3. No one else can access your files directly.

How CloudFront URL Works

CloudFront URL is simply your CloudFront domain combined with the S3 file path. You build this URL yourself during the import process and store it in the database.

```
S3 file path:       audio/surah_1.mp3
CloudFront domain:  https://d1234abcd.cloudfront.net
CloudFront URL:     https://d1234abcd.cloudfront.net/audio/surah_1.mp3
```

4. JSON Data Format (From Company)

The company will provide JSON files containing all Islamic content. Below is the expected structure:

4.1 Quran Data (quran.json)

```
{
  "surahs": [
    {
      "number": 1,
      "name_arabic": "الفاتحة",
      "name_english": "Al-Fatiha",
      "audio_url": "https://theirserver.com/audio/surah_1.mp3",
      "verses": [
        {
          "number": 1,
          "arabic_text": "بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ",
          "translation": "In the name of Allah, the Entirely Merciful..."
        },
        {
          "number": 2,
          "arabic_text": "الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ",
          "translation": "All praise is due to Allah, Lord of the worlds"
        }
      ]
    }
  ]
}
```

4.2 Duas Data (duas.json)

```
{
  "duas": [
    {
      "id": 1,
      "title": "Dua before eating",
      "arabic_text": "بِسْمِ اللَّهِ",
      "translation": "In the name of Allah",
      "audio_url": "https://theirserver.com/audio/dua_1.mp3",
      "category": "daily"
    }
  ]
}
```

4.3 Tasbihaat Data (tasbihaat.json)

```json
{
  "tasbihaat": [
    {
      "id": 1,
      "arabic_text": "سُبْحَانَ اللهِ",
      "translation": "Glory be to Allah",
      "count": 33,
      "audio_url": "https://theirserver.com/audio/tasbih_1.mp3"
    }
  ]
}
```

5. Database Design

Aurora MySQL stores all text data and CloudFront URLs. Actual audio/image files are never stored in the database.

5.1 Migrations

surahs table

```
Schema::create('surahs', function (Blueprint $table) {
    $table->id();
    $table->integer('number')->unique();
    $table->string('name_arabic');
    $table->string('name_english');
    $table->string('audio_url');  // CloudFront URL stored here
    $table->timestamps();
});
```

verses table

```
Schema::create('verses', function (Blueprint $table) {
    $table->id();
    $table->foreignId('surah_id')->constrained()->onDelete('cascade');
    $table->integer('verse_number');
    $table->text('arabic_text');
    $table->text('translation');
    $table->timestamps();
});
```

duas table

```
Schema::create('duas', function (Blueprint $table) {
    $table->id();
    $table->string('title');
    $table->text('arabic_text');
    $table->text('translation');
    $table->string('audio_url');  // CloudFront URL
    $table->string('category')->nullable();
    $table->timestamps();
});
```

5.2 Models & Relationships

```
// Surah Model
class Surah extends Model {
    protected $fillable = ['number', 'name_arabic', 'name_english', 'audio_url'];

    public function verses() {
        return $this->hasMany(Verse::class);
    }
}

// Verse Model
class Verse extends Model {
    protected $fillable = ['surah_id', 'verse_number', 'arabic_text',
'translation'];

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

6. Import Command (One Time)

The import command reads JSON data, uploads media files to S3, builds CloudFront URLs, and saves everything to the database. This runs only once during initial setup.

6.1 Install Laravel S3 Package

```
composer require league/flysystem-aws-s3-v3
```

6.2 Add AWS Config to .env

```
AWS_ACCESS_KEY_ID=your-iam-access-key
AWS_SECRET_ACCESS_KEY=your-iam-secret-key
AWS_DEFAULT_REGION=ap-south-1
AWS_BUCKET=islam360-content
CLOUDFRONT_URL=https://d1234abcd.cloudfront.net
```

6.3 Full Import Command

```
<?php
namespace App\Console\Commands;

use Illuminate\Console\Command;
use Illuminate\Support\Facades\Storage;
use App\Models\Surah;
use App\Models\Verse;

class ImportQuranData extends Command
{
    protected $signature = 'import:quran';
    protected $description = 'Import Quran data from JSON file';

    public function handle()
    {
        $data = json_decode(
            file_get_contents(storage_path('app/quran.json')), true
        );

        foreach ($data['surahs'] as $surahData) {

            // Step 1: Download audio from company URL
            $audioContent = file_get_contents($surahData['audio_url']);

            // Step 2: Upload audio to your S3 bucket
            $s3Path = 'audio/surah_' . $surahData['number'] . '.mp3';
            Storage::disk('s3')->put($s3Path, $audioContent);
            // Step 3: Build CloudFront URL
            $cloudfrontUrl = env('CLOUDFRONT_URL') . '/' . $s3Path;
            // Result: https://d1234abcd.cloudfront.net/audio/surah_1.mp3

            // Step 4: Save surah with CloudFront URL to database
            $surah = Surah::create([
                'number'       => $surahData['number'],
                'name_arabic'  => $surahData['name_arabic'],
                'name_english' => $surahData['name_english'],
                'audio_url'    => $cloudfrontUrl,
            ]);

            // Step 5: Save all verses for this surah
            foreach ($surahData['verses'] as $verseData) {
                Verse::create([
                    'surah_id'     => $surah->id,
                    'verse_number' => $verseData['number'],
                    'arabic_text'  => $verseData['arabic_text'],
                    'translation'  => $verseData['translation'],
                ]);
            }

            $this->info('Imported Surah ' . $surahData['number']);
        }

        $this->info('Quran import complete!');
    }
}
```

6.4 Run the Import

```
# Place JSON files in storage/app/ directory
# Then run:
php artisan import:quran
php artisan import:duas
php artisan import:tasbihaat
```

ℹ This import runs ONCE only during initial setup. After this the database has all content and CloudFront has all media files. The API just reads from them at runtime.

7. Laravel API

The API reads from Aurora MySQL and returns JSON including CloudFront URLs. Laravel never fetches or serves audio/image files directly.

7.1 Routes

```
// routes/api.php
Route::get('/surahs', [SurahController::class, 'index']);
Route::get('/surahs/{number}', [SurahController::class, 'show']);
Route::get('/duas', [DuaController::class, 'index']);
Route::get('/tasbihaat', [TasbihController::class, 'index']);
```

7.2 Controller

```
class SurahController extends Controller
{
    public function index()
    {
        $surahs = Surah::select('id', 'number', 'name_arabic', 'name_english',
'audio_url')
                       ->get();
        return response()->json($surahs);
    }

    public function show($number)
    {
        $surah = Surah::with('verses')
                      ->where('number', $number)
                      ->firstOrFail();
        return response()->json($surah);
    }
}
```

7.3 API Response Example

When mobile app calls GET /api/surahs/1 it receives:

```
{
  "id": 1,
  "number": 1,
  "name_arabic": "الفاتحة",
  "name_english": "Al-Fatiha",
  "audio_url": "https://d1234abcd.cloudfront.net/audio/surah_1.mp3",
  "verses": [
    {
      "verse_number": 1,
      "arabic_text": "بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ",
      "translation": "In the name of Allah, the Entirely Merciful..."
    },
    {
      "verse_number": 2,
      "arabic_text": "الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ",
      "translation": "All praise is due to Allah, Lord of the worlds"
    }
  ]
}
```

The mobile app then plays the audio by hitting the audio_url directly from CloudFront. Laravel is not involved in that at all.

8. Complete Flow Summary

8.1 Import Phase (One Time Only)

```
Company JSON File
      │
      └──▶ Laravel Import Command
                │
                ├──▶ Text data (arabic, translation) ──▶ Aurora MySQL
                │
                └──▶ Media files (mp3, images) ──▶ Upload to S3
                                                       │
                                                  Build CloudFront URL
                                                       │
                                                  Save URL in database
```

8.2 Runtime Phase (Every API Call)

```
Mobile App
      │
      ├──▶ GET /api/surahs/1
      │          │
      │    Laravel API (EC2)
      │          │
      │    Aurora MySQL ──▶ Returns text + CloudFront URL
      │          │
      │    JSON Response to Mobile App
      │
      └──▶ Mobile App plays audio
                │
           CloudFront URL directly
                │
           CloudFront Edge (nearest location)
                │
           S3 Bucket (if not cached)
```

ℹ Laravel never touches audio or image files at runtime. It only returns the CloudFront URL as a string. The mobile app handles media playback directly from CloudFront.

8.3 AWS Infrastructure Summary

Service	Purpose

EC2 (t3.small)	Runs Laravel API inside Docker container
Aurora MySQL Serverless v2	Stores all text data and CloudFront URLs
S3 Bucket (private)	Stores all audio, image, PDF files
CloudFront CDN	Delivers S3 files globally from nearest edge location
IAM User	Laravel uses this to upload files to S3

9. Complete .env Configuration

```
APP_NAME=Islam360
APP_ENV=production
APP_KEY=base64:your-key-here
APP_DEBUG=false
APP_URL=https://api.islam360.com

# Database - Aurora MySQL
DB_CONNECTION=mysql
DB_HOST=your-aurora-endpoint.rds.amazonaws.com
DB_PORT=3306
DB_DATABASE=islam360
DB_USERNAME=your_user
DB_PASSWORD=your_password

# AWS S3
AWS_ACCESS_KEY_ID=your-iam-access-key
AWS_SECRET_ACCESS_KEY=your-iam-secret-key
AWS_DEFAULT_REGION=ap-south-1
AWS_BUCKET=islam360-content
AWS_USE_PATH_STYLE_ENDPOINT=false

# CloudFront
CLOUDFRONT_URL=https://d1234abcd.cloudfront.net

# Filesystem
FILESYSTEM_DISK=s3
```

10. Deployment Checklist

AWS Setup

- S3 bucket created with public access blocked
- IAM user created with S3FullAccess policy
- IAM access keys saved securely in .env
- CloudFront distribution created with OAC
- S3 bucket policy updated to allow CloudFront only
- CloudFront domain copied and added to .env as CLOUDFRONT_URL

Laravel Setup

- league/flysystem-aws-s3-v3 package installed
- All .env values configured correctly
- Migrations run: php artisan migrate
- Import commands run for all content types
- Verify CloudFront URLs are saved in database
- Test API endpoint returns correct JSON with CloudFront URLs

Verification

- Hit CloudFront URL in browser — audio/image should load
- Call API endpoint — verify arabic text and audio_url in response
- Mobile app plays audio from CloudFront URL directly